VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODELOVÁNÍ ELEKTRICKÝCH OBVODŮ VE SPECIALIZOVANÉM PARALELNÍM SYSTÉMU

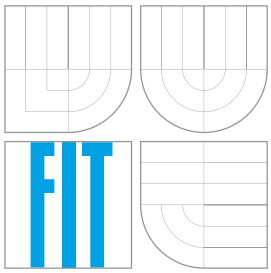
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ROMAN JANKO

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODELOVÁNÍ ELEKTRICKÝCH OBVODŮ VE SPECIALIZOVANÉM PARALELNÍM SYSTÉMU

ELECTRIC CIRCUITS SIMULATIONS IN A SPECIALIZED PARALLEL SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ROMAN JANKO

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. JIŘÍ KUNOVSKÝ, CSc.

BRNO 2013

Abstrakt

V práci je uveden přehled metod pro numerické řešení diferenciálních rovnic. Dále možnosti jejich paralelizace, tedy rozdělení výpočetních operací na více mikroprocesorů, s důrazem kladeným na použití metody Taylorovy řady. Další část se věnuje popisu specializovaného paralelního systému, který byl vyvinut pro rychlé řešení soustav těchto rovnic. Diferenciální rovnice jsou vhodným způsobem pro popis elektrických obvodů. Důležitou vlastností každého obvodu je jeho chování ve frekvenční oblasti. Cílem práce bylo navrhnout a implementovat program, který bude vyšetřovat frekvenční charakteristiky střídavých elektrických obvodů. Je prezentována vlastní metoda analyzující obvod a automaticky k němu sestavující příslušné rovnice, které jsou následně vyřešeny v systému TKSL. V závěru je zhodnocena časová náročnost výpočtu v porovnání s programem Matlab.

Abstract

This work provides an overview of methods for the numerical solution of differential equations. Options of their parallelization, a division of computational operations on multiple microprocessors, are provided with emphasis placed on the Taylor series. The next part of the work is devoted to the description of a specialized parallel system, which was design to fast solving of these equations. Differential equations are appropriate to describe electrical circuits. An important characteristic of each circuit is its behavior in the frequency domain. The aim of this thesis was to design and implement a program which investigate frequency characteristics of AC circuits. A method for analyzing a circuit and automatically assembling corresponding equations is presented. These differential equations are then solved in TKSL. At the end of this work a time consumption is evaluated and compared with Matlab.

Klíčová slova

numerická integrace, diferenciální rovnice, Taylorova řada, paralelní systém, frekvenční charakteristiky, STA, MNA, TKSL, Matlab, SCAM

Keywords

numerical integration, differential equations, Taylor series, parallel system, frequency characteristics, STA, MNA, TKSL, Matlab, SCAM

Citace

Roman Janko: Modelování elektrických obvodů ve specializovaném paralelním systému, diplomová práce, Brno, FIT VUT v Brně, 2013

Modelování elektrických obvodů ve specializovaném paralelním systému

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně pod vedením doc. Ing. Jiřího Kunovského, CSc. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Janko
12. května 2013

Poděkování

Děkuji doc. Ing. Jiřímu Kunovskému, CSc. za jeho odborné vedení, podnětné připomínky a rady.

Obsah

1 Úvod **4**

2 Numerické řešení diferenciálních rovnic **6**
2.1 Jednokrokové metody . . . 7
2.1.1 Eulerova metoda . . . 7
2.1.2 Metoda Runge-Kutta . . . 8
2.1.3 Taylorova řada . . . 8
2.2 Vícekrokové metody . . . 8
2.3 Řešení rovnic vyšších řádů . . . 9
2.3.1 Metoda snižování řádu derivace . . . 9
2.3.2 Metoda postupné integrace . . . 10
2.4 Řešení soustav diferenciálních rovnic . . . 11

3 Paralelní numerické řešení diferenciálních rovnic **12**
3.1 Převod rovnic na blokové schéma . . . 12
3.2 Modifikovaná metoda Taylorovy řady . . . 13
3.3 Tvořící diferenciální rovnice . . . 15

4 Specializovaný paralelní systém **17**
4.1 Numerické integrátory . . . 17
4.1.1 Paralelně-paralelní (P-P) integrátor . . . 18
4.1.2 Sériově-paralelní (S-P) integrátor . . . 19
4.1.3 Sériově-sériový (S-S) integrátor . . . 20
4.2 Řídící jednotka . . . 21
4.3 Propojovací síť . . . 22

5 Simulace obvodů **23**
5.1 Součástky . . . 23
5.1.1 Pasivní součástky . . . 23
5.1.2 Nezávislé zdroje . . . 24
5.2 Sestavení rovnic . . . 24
5.3 Reprezentace obvodu . . . 25
5.4 Modifikovaná metoda uzlových napětí . . . 25
5.4.1 Ukázkový příklad . . . 26
5.4.2 Omezení metody . . . 27

6 Návrh **28**
6.1 Specifikace požadavků . . . 28
6.2 Návrh programu . . . 29
6.3 Modul pro editaci schémat . . . 30
6.4 Modul pro simulaci . . . 31
6.5 Modul analyzující obvod . . . 31
6.6 Modul pro zpracovaní výsledků simulace . . . 31
6.7 Modul pro zobrazení výsledků . . . 32

7 Implementace **33**
7.1 Použité nástroje . . . 33
7.2 Analyzátor obvodu . . . 33
7.2.1 Použitá metoda pro sestavení rovnic . . . 33
7.2.2 Měření na osciloskopech . . . 36
7.2.3 Zkratované součástky . . . 36
7.3 Simulátor řešící sestavené rovnice . . . 36
7.3.1 Napojení na TKSL/C . . . 37
7.3.2 Ustálení výstupního napětí . . . 37
7.3.3 Volba integračního kroku a lokalizace amplitudy . . . 37
7.3.4 Zpracování dat z TKSL/C . . . 38
7.4 Koeficienty zrychlení . . . 39

8 Testování **41**
8.1 Manuální testovaní . . . 41
8.2 Automatické testování . . . 42

9 Zhodnocení časové náročnosti **43**
9.1 Vyšetřování frekvenčních charakteristik v Matlabu . . . 43
9.2 Srovnání časové náročnosti s programem Matlab . . . 44

10 Závěr **46**

Literatura **49**

A Obsah CD **50**

B Manuál k programu **51**
B.1 Požadavky . . . 51
B.2 Překlad . . . 51
B.3 Ovládání a popis programu . . . 52
B.4 Použití testovacího programu . . . 54

Seznam obrázků

2.1 Přesné a přibližné řešení diferenciální rovnice 6

3.1 Blokové schéma rovnic $y'' = C$ a $y'' + k.y = C$ 13
3.2 Blokové schéma pro soustavu rovnic 14
3.3 Blokové schéma rovnice před a po transformaci 16

4.1 Blokové schéma specializovaného paralelního systému 17
4.2 Blokové schéma paralelně-paralelního (P-P) integrátoru 18
4.3 Blokové schéma sériově-paralelního (S-P) integrátoru 19
4.4 Blokové schéma sériově-sériového (S-S) integrátoru 20
4.5 Blokové schéma řadiče 21
4.6 Propojovací síť se třemi integrátory a jednou sčítačkou 22

5.1 Jednoduchý lineární obvod 24
5.2 Grafová reprezentace obvodu 25
5.3 Obvod s rezistory 26

6.1 Návrh programu 30
6.2 Editor schémat 31

7.1 RLC obvod 34
7.2 Neustálený signál 37
7.3 Problém nalezení maxima 38
7.4 Možnosti odečtení fázového posunu 39

8.1 Výstup z programu provádějící automatické testy 41

9.1 Srovnání s programem Matlab 45

B.1 Editor elektrických schémat 52
B.2 Nastavení parametrů simulace a její spuštění 53
B.3 Výsledky simulace 53

Kapitola 1

Úvod

Pod pojmem modelování rozumíme cílevědomou činnost, která slouží k získávání informací o jednom systému prostřednictvím jiného systému tzv. modelu. Je-li modelovaný systém jednoduchý, nebo můžeme-li formulovat tak zjednodušující předpoklady, aby byl model řešitelný analyticky, popíšeme chování systému matematickými vztahy a hledané veličiny stanovujeme matematickými prostředky. Hlavní předností analytického řešení je menší časová náročnost řešení matematického modelu. Jde však o modely jednoduché nebo podstatně zjednodušené. V současné době je však stále aktuálním problémem analýza složitých systémů, jejichž specifickými vlastnostmi jsou velká rozsáhlost, velká dynamičnost probíhajících procesů a složitý charakter vztahů mezi prvky systému. Komplexní analýzu těchto systémů umožnil teprve rozvoj číslicových počítačů, které jsou dnes nejperspektivnějším prostředkem pro studium složitých dynamických systémů prostřednictvím jejich modelů. Proces modelování systémů na počítačích můžeme velmi zjednodušeně rozdělit do tří základních etap - vytvoření abstraktního modelu, vytvoření simulačního modelu a samotná simulace. Simulací označujeme etapu experimentování s reprezentací simulačního modelu. Jejím cílem je analýza chování systému v závislosti na vstupních veličinách a na hodnotách parametrů. [25]

Simulace na číslicových počítačích je velice rozšířená disciplína. Uplatnění nachází ve velkém množství oborů jako medicína, fyzika, meteorologie, ekonomika, technika. V oblasti elektrotechniky hraje simulace klíčovou roli především v kontrole a ověření správnosti návrhu elektrických a elektronických systémů a to před jejich vlastní výrobou a nasazením. Používá se v širokém spektru aplikací, od integrovaných obvodů a mikroelektroniky po elektrické distribuční sítě. Techniky pro simulaci obvodů byly poprvé představeny v 50. a 60. letech minulého století. Jedním z prvních rozšířených nástrojů pro simulaci obvodů byl program SPICE uvedený roku 1973.

Důležitou vlastností každého obvodu je jeho chování ve frekvenční oblasti. Cílem práce je proto navrhnout a implementovat program, který bude vyšetřovat frekvenční charakteristiky střídavých elektrických obvodů. Frekvenční charakteristiky jsou souhrnným označením pro amplitudovou a fázovou charakteristiku. První jmenovaná zachycuje závislost přenosu, což je poměr velikosti amplitud výstupního a vstupního napětí, na frekvenci. Druhá charakteristika zachycuje závislost velikosti fázového posunu mezi těmito napětími na frekvenci. Modelovanými obvody budou tzv. pasivní kmitočtové filtry. Tedy obvody obsahující pouze pasivní součástky, tj. rezistory, kondenzátory a cívky.

Jedním ze způsobů, jakým modelovat chování elektrických obvodů jsou diferenciální rovnice. Pro jejich řešení existují systémy jako TKSL, Matlab či Maple. TKSL je založeno na modifikované metodě Taylorovy řady a vyznačuje se vysokou přesností a rychlostí. V bu-

doucnu bude možné výpočet v TKSL ještě více urychlit díky specializovanému paralelnímu systému na čipu FPGA. Nyní bohužel chybí vazba mezi oběma zmiňovanými systémy. Tato vazba je předmětem aktivního vývoje.

Text práce je členěn do několika kapitola. V kapitole 2 jsou popsány metody pro řešení diferenciálních rovnic a jejich soustav. Kapitola 3 se zabývá možnostmi jejich paralelního výpočtu. Kapitola 4 se věnuje popisu specializovaného paralelního systému, který byl navrhnut Ing. Michalem Krausem pro rychlé řešení diferenciálních rovnic na FPGA. Nejdůležitějšími bloky paralelního systému jsou numerické integrátory, tj. P-P (paralelně-paralelní), S-P (sériově-paralelní) a S-S (sériově-sériové) integrátory. Kapitola 5 se zabývá simulací elektrických obvodů a hlavně porovnává známé metody pro automatické sestavení rovnic k danému obvodu. Kapitola 6 specifikuje požadavky na vytvářený program a popisuje jeho samotný návrh. Kapitola 7 se zabývá implementací programu, popisem použitých nástrojů a vytvořené metody sestavující automaticky diferenciální rovnice. Dále je popsáno řešení problémů jako ustálení výstupního napětí a lokalizace maxima. Kapitola 8 popisuje, jakým způsobem byla aplikace otestována. Poslední kapitola 9 hodnotí časovou složitost v porovnání se systémem Matlab. Srovnání bylo provedeno pomocí sady elektrických obvodů z výuky ITO, IPR a VNV. Příloha obsahuje návod na použití vytvořené aplikace a spuštění testovacího programu.

Kapitoly 2, 3 a 4 byly zpracovány v rámci semestrálního projektu. Jak již bylo řečeno, kapitoly 2 a 3 uvádějí problematiku řešení diferenciálních rovnic do širšího kontextu. Kapitola 4 popisuje specializovaný paralelní systém. Výsledky z této kapitoly budou použity pro výpočet koeficientů zrychlení.

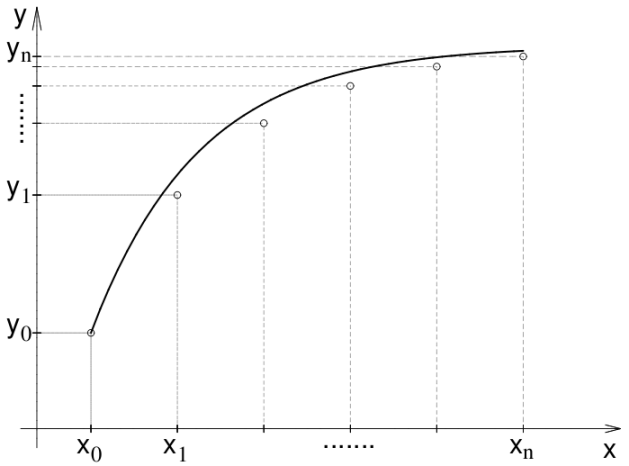
Kapitola 2

Numerické řešení diferenciálních rovnic

Diferenciální rovnice jsou jednou z možností popisu elektronických obvodů. Můžeme je řešit analyticky nebo numericky. Analytické metody jsou v praxi málo využitelné, protože řešení technických úloh vede na velmi složité soustavy diferenciálních rovnic. S rozvojem výpočetní techniky je proto nahradily metody numerické. Nedosahují takové přesnosti jako analytické metody a při jakékoliv změně parametrů je nutné provést celý výpočet znovu, ale umožňují řešit i velmi složité soustavy diferenciálních rovnic.

Společným znakem všech dále uvedených metod je, že se řešení nehledá jako spojitá funkce definovaná na celém zkoumaném intervalu $\langle a, b\rangle$, ale hodnoty přibližného řešení se počítají pouze v konečném počtu bodů $a = x_0 < x_1 < ... < x_n = b$. Těmto bodům se říká *uzlové body*. Rozdíl $h_i = x_{i+1} - x_i$ se nazývá *krok*. Na velikost kroku záleží jak moc přesně se přiblížíme k řešení. Chceme-li znát přibližnou hodnotu řešení v jiném než uzlovém bodě, můžeme použít některou z interpolačních metod, např. nahradit řešení lomenou čarou procházející uzlovými body. [4]

Na obrázku 2.1 je vidět přesné řešení, které je vykresleno plnou čarou a přibližné hodnoty řešení v uzlových bodech - vyznačeno kroužky.



Obrázek 2.1: Přesné a přibližné řešení diferenciální rovnice, Zdroj: [4]

Numerické metody pro řešení diferenciálních rovnic dělíme na dva typy:

1. **Jednokrokové metody** počítají přibližnou hodnotu řešení v dalším uzlovém bodě pomocí hodnoty v předchozím uzlovém bodě.
2. **Vícekrokové metody** počítají přibližnou hodnotu řešení v dalším uzlovém bodě pomocí hodnot v několika předchozích uzlových bodech.

2.1 Jednokrokové metody

Jednokrokové metody počítají přibližnou hodnotu řešení v dalším uzlovém bodě pomocí hodnoty v předchozím uzlovém bodě. Umožňují dynamickou změnu integračního kroku. Mezi jednokrokové metody patří Eulerova metoda, metoda Runge-Kutta a Taylorova metoda.

2.1.1 Eulerova metoda

Jedná se o nejjednodušší jednokrokovou metodu. Pro výpočet používá pouze první dva členy Taylorovy řady.

Mějme danou obyčejnou diferenciální rovnici prvního řádu s počáteční podmínkou

$$y' = f(x, y), \qquad y(x_0) = y_0 \tag{2.1}$$

a pravidelnou síť $\{x_0, x_1, \ldots, x_n\}$ s krokem h. Hodnota $y(x_0)$ je přesné řešení v bodě x_0. Z počáteční úlohy 2.1 vyplývá, že ve všech bodech rovnice platí

$$y'(x_i) = f(x_i, y(x_i)) \tag{2.2}$$

Derivaci na levé straně nahradíme diferencí. Dostaneme

$$\frac{y(x_{i+1}) - y(x_i)}{h} \doteq f(x_i, y(x_i)) \tag{2.3}$$

Nahradíme-li $y(x_i)$ přibližnou hodnotou y_i, můžeme odtud vyjádřit přibližnou $y(x_{i+1})$ jako

$$y_{i+1} = y_i + hf(x_i, y_i), \tag{2.4}$$

kde y_i je přibližné řešení v předchozím uzlovém bodě,
h označuje integrační krok.

Hodnotu řešení v bodě x_0 známe z počáteční podmínky. Rozumným zkracováním kroku lze dosáhnout vyšší přesnosti. Je nutné brát v úvahu, že výpočet každého kroku vychází z hodnot kroku předešlého, který je již zatížen chybou z předchozích výpočtů. Proto nemusíme extrémně malým krokem dosáhnout vyšší přesnosti. [9]

2.1.2 Metoda Runge-Kutta

Jednokroková metoda, která realizuje výpočet i mezi jednotlivými uzly. Existuje několik variant metod Runge-Kutta, které se liší přesností a stabilitou. Nejčastěji používanou je metoda Runge-Kutta 4. řádu.

Vztah pro výpočet následujícího uzlového bodu metodou Runge-Kutta 4. řádu je:

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.5}$$
$$k_1 = f(x_n, y_n) \tag{2.6}$$
$$k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \tag{2.7}$$
$$k_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \tag{2.8}$$
$$k_4 = f(x_n + h, y_n + hk_3) \tag{2.9}$$

kde y_n je přibližné řešení v předchozím uzlovém bodě,
h označuje integrační krok,
f je derivace v předchozím uzlovém bodě a
k_1, k_2, k_3, k_4 jsou pomocné výpočty prováděné uvnitř kroku.

Metoda Runge-Kutta provádí uvnitř kroku další pomocné výpočty, čímž dosahuje vyšší přesnosti. Při stejné délce kroku je podstatně přesnější než Eulerova metoda. [9]

2.1.3 Taylorova řada

Taylorova řada je jednokroková metoda, která se vyjadřuje zápisem:

$$y_{i+1} = y_i + \frac{h}{1!}f'(y_i) + \frac{h^2}{2!}f''(y_i) + \frac{h^3}{3!}f'''(y_i) + \cdots + \frac{h^n}{n!}f^{(n)}(y_i) \tag{2.10}$$

kde h je integrační krok.

Metoda Taylorovy řady umožňuje určit přesnost výpočtu. Výpočet se ukončí ve chvíli, kdy rozdíl dvou sousedních členů je menší než požadovaná přesnost. Čím více členů použijeme, tím je metoda přesnější. Nevýhodou je nutnost použití vyšších derivací.

2.2 Vícekrokové metody

Vícekrokové metody počítají přibližnou hodnotu řešení v dalším uzlovém bodě pomocí hodnot v několika předchozích uzlových bodech.

Obecně má k-kroková metoda tvar

$$y_{n+1} = \sum_{i=1}^{k} a_i y_{n-i+1} + h\sum_{i=0}^{k} b_i f(x_{n+1-i}, y_{n+1-i}) \tag{2.11}$$

kde k je přirozené číslo,
a_i, b_i jsou konstanty (alespoň jedna z nich musí být nenulová),

Při startu výpočtu je nutné použít některou jednokrokovou metodu, pomocí které vypočítáme prvních k hodnot potřebných k výpočtu první hodnoty vícekrokovou metodou. Tato vlastnost velmi komplikuje použití v simulacích při častých změnách integračního kroku. Vícekrokové metody jsou přesnější než jednokrokové metody, ale jejich výpočet je náročnější. Příkladem vícekrokové metody je metoda Adams-Bashforth. [9]

2.3 Řešení rovnic vyšších řádů

Rovnice vyšších řádů musíme převést na soustavu rovnic prvního řádu, protože máme vhodné numerické metody pouze pro řešení rovnic prvního řádu. Každou rovnici vyššího řádu je možné transformovat na soustavu rovnic prvního řádu.

Zaměříme se na dvě základní metody:

- **Metoda snižování řádu derivace** je jednodušší, ale vyžaduje, aby na pravé straně rovnice nebyly derivace vstupu.
- **Metoda postupné integrace** zvládne i rovnice s derivacemi vstupu na pravé straně.

Obě metody využívají úprav rovnice a zavádění pomocných proměnných. [21]

2.3.1 Metoda snižování řádu derivace

Pro převod rovnice ve tvaru

$$ay^{(n)} + by^{(n-1)} + \cdots + ky' + zy = f(x, y, y', \dots) \tag{2.12}$$

postupujeme následovně:

1. Osamostatníme nejvyšší řád derivace na levou stranu.
$$y^{(n)} = \dots$$
2. Postupným vkládáním integrátorů získáme y.
$$\begin{aligned} y^{(n-1)} &= \int y^{(n)} \mathrm{d}t & (2.13)\\ y^{(n-2)} &= \int y^{(n-1)} \mathrm{d}t & (2.14)\\ &\dots \\ y &= \int y' \mathrm{d}t & (2.15) \end{aligned}$$

Podmínkou použití je, že na pravé straně nesmí být žádné derivace vstupů $(x', x'', \dots)$.

Příklad

Mějme diferenciální rovnici druhého řádu $y'' - 3y' - 2y = 8x$, nulové počáteční podmínky. Po úpravě metodou snižování řádu derivace získáme soustavu rovnic

$$\begin{aligned} y'' &= 3y' + 2y + 8x & (2.16)\\ y' &= \int y'' \mathrm{d}t & (2.17)\\ y &= \int y' \mathrm{d}t & (2.18) \end{aligned}$$

2.3.2 Metoda postupné integrace

Tato metoda je vhodná i pro rovnice s derivacemi vstupů x. Pro převod rovnice ve tvaru

$$ay^{(n)} + by^{(n-1)} + \cdots + ky' + zy = f(x, x', x'', \ldots, y, y', \ldots) \tag{2.19}$$

postupujeme takto:

1. Osamostatníme nejvyšší řád derivace na levou stranu.

   $$y^{(n)} = \ldots$$

2. Zintegrujeme

   $$y^{(n-1)} = vyraz1 + \int vyraz2\mathrm{d}t \tag{2.20}$$

   a zavedeme pomocné proměnné Ai pro výrazy, kde zůstal integrál

   $$A1 = \int vyraz2\mathrm{d}t \tag{2.21}$$

   Po substituci dostáváme

   $$y^{(n-1)} = vyraz1 + A1 \tag{2.22}$$

   a postupně integrujeme, dokud nemáme rovnici pro výpočet y.

3. Nakonec je nutné vypočítat nové počáteční podmínky pro všechny pomocné proměnné, které jsme zavedli.

   $$A1(0) = y^{(n-1)}(0) - vyraz1(0) \tag{2.23}$$

   Hodnotu $y^{(n-1)}(0)$ známe z původních počátečních podmínek.

Metoda postupné integrace je použitelná pouze pro konstantní koeficienty a nejvyšší řád derivace x musí být menší nebo roven než nejvyšší řád derivace y.

Příklad

Máme rovnici $y'' + 2y' + y = 3x' + x$ a nulové počáteční podmínky. Rovnici převedeme do operátorového tvaru

$$p^2y + 2py + y = 3px + x \tag{2.24}$$

a postupně upravujeme

$$p^2y = p(3x - 2y) + (x - y) \tag{2.25}$$

zintegrujeme

$$py = (3x - 2y) + \frac{1}{p}(x - y) \tag{2.26}$$

zavedeme pomocnou proměnnou $A1$

$$py = (3x - 2y + A1) \tag{2.27}$$

znovu zintegrujeme

$$y = \frac{1}{p}(3x - 2y + A1) \tag{2.28}$$

zavedeme pomocnou proměnnou $A2$

$$y = A2 \tag{2.29}$$

Nyní už máme výslednou soustavu rovnic:

$$\begin{aligned} y &= A2 && (2.30)\\ A2 &= \frac{1}{p}(3x - 2y + A1) && (2.31)\\ A1 &= \frac{1}{p}(x - y) && (2.32) \end{aligned}$$

Zbývá jen vypočítat nové počáteční podmínky.

2.4 Řešení soustav diferenciálních rovnic

Soustavy diferenciálních rovnic prvního řádu se řeší velmi podobně jako jediná rovnice, až na to, že místo jediné funkce f a skaláru y_i pracujeme s vektory funkcí a hodnot řešení.

Mějme soustavu rovnic

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \ldots, y_n) \quad y_1(x_0) = \eta_1 \\ y_2' &= f_2(x, y_1, y_2, \ldots, y_n) \quad y_2(x_0) = \eta_2 \\ &\vdots \qquad\qquad\qquad\quad \vdots \\ y_n' &= f_n(x, y_1, y_2, \ldots, y_n) \quad y_n(x_0) = \eta_n \end{aligned} \tag{2.33}$$

Soustavu můžeme vektorově přepsat jako

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \quad \mathbf{y}(x_0) = \boldsymbol{\eta} \tag{2.34}$$

kde $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$, $\mathbf{f} = (f_1, f_2, \ldots, f_n)^T$ a $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_n)^T$. Pro její numerické řešení můžeme použít kteroukoliv z dříve popsaných metod, jen je potřeba pracovat s vektory. [4]

Kapitola 3

Paralelní numerické řešení diferenciálních rovnic

Přestože myšlenka paralelního výpočtu a paralelního spojení velkého počtu mikroprocesorů je zajímavá, není jednoduché dosáhnout velkého nárůstu výkonnosti. Hlavním problémem je, že většina algoritmů nebyla vyvíjena pro paralelní systémy a i malé procento sekvenčních kroků může vést k velké redukci výkonu celého systému.

Idea paralelního numerické řešení diferenciálních rovnic pochází z dob analogových počítačů, kdy se podle blokového schématu provedlo zapojení jednotlivých prvků a výpočet probíhal paralelně. Abychom byli schopni aplikovat tento postup na diskrétní počítače, je nutné realizovat elementární prvky jako diskrétní výpočetní jednotky. Elementárními prvky jsou:

- konstanty (bez vstupu),
- násobičky konstantou,
- invertory,
- základní aritmetické operace +, -, *, /,
- integrátory,
- zpoždění.

3.1 Převod rovnic na blokové schéma

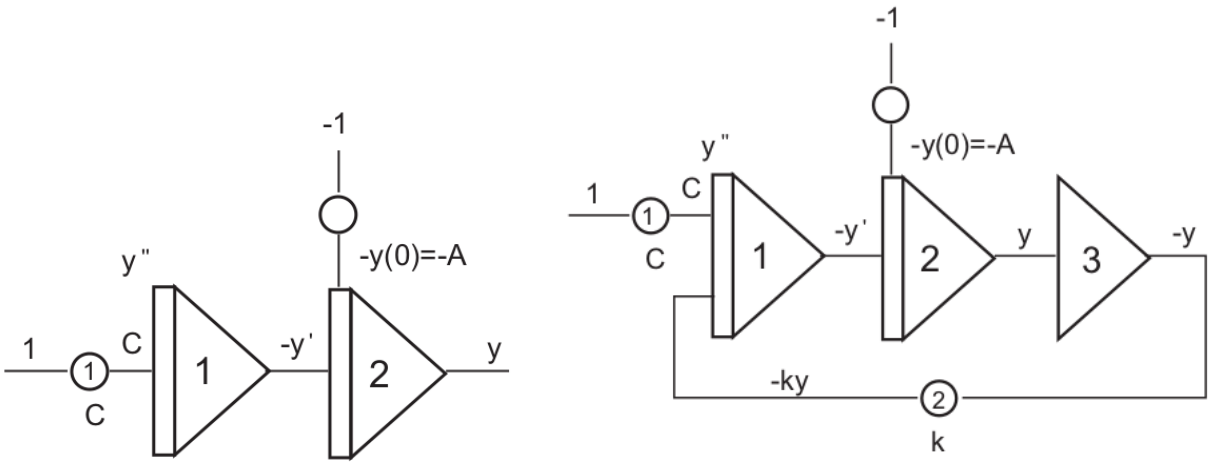
Popis pomocí blokového schéma vychází z postupů používaných na analogových počítačích. Blokové schéma je velmi názorné. Pro větší soustavy rovnic je vhodné schéma hierarchicky strukturovat kvůli zachování přehlednosti.

Mějme rovnice

$$y'' = C, \qquad y(0) = A, \quad y'(0) = 0, \tag{3.1}$$

$$y'' + k.y = C, \qquad y(0) = A, \quad y'(0) = 0, \tag{3.2}$$

Začneme s převodem rovnice 3.1. Na vstup integrátoru 1 s počáteční podmínkou 0 připojíme násobičku konstantou s hodnotou nastavenou na C. C je podle rovnice rovno



Obrázek 3.1: Blokové schéma rovnic $y'' = C$ a $y'' + k.y = C$, Zdroj: [16]

y''. Výstup integrátoru 1 je $-y'$, protože se jedná o invertující integrátor. Výstup $-y'$ je přiveden na vstup integrátoru 2, jehož výstup y odpovídá požadovanému řešení.

Princip vytvoření blokového schéma pro rovnici 3.2 je obdobný. Všimněme si, že schéma obsahuje zpětnou vazbu. Ve zpětné vazbě je zapojena násobička konstantou s hodnotou nastavenou na k. Prvek s označením 3 je invertor.

Je důležité si uvědomit, že v obou případech pracují všechny elementární prvky (integrátory, invertor, násobičky konstantou, ...) paralelně.

3.2 Modifikovaná metoda Taylorovy řady

Modifikovaná metoda Taylorovy řady (viz [12]) pro numerické řešení obyčejných diferenciálních rovnic byla vytvořena na základě analýzy nejjednodušší homogenní lineární diferenciální rovnice 1. řádu s konstantními koeficienty:

$$y' = y, \qquad y(0) = y_0, \tag{3.3}$$

Protože platí $y' = y'' = y''' = \ldots$, tak po aplikaci Taylorovy řady dostáváme:

$$y_1 = y_0 + h \cdot y_0 + \frac{h^2}{2!} y_0 + \frac{h^3}{3!} y_0 + \ldots \frac{h^n}{n!} y_0 \tag{3.4}$$

Provedeme-li v rovnici 3.4 následující substituce

$$y_1 = y_0 + DY1_0 + DY2_0 + DY3_0 + \ldots DYp_0 \tag{3.5}$$

$$DY1_0 = h \cdot y_0 \tag{3.6}$$

$$DY2_0 = \frac{h}{2} DY1_0 \tag{3.7}$$

$$DY3_0 = \frac{h}{3} DY2_0 \tag{3.8}$$

$$\vdots$$

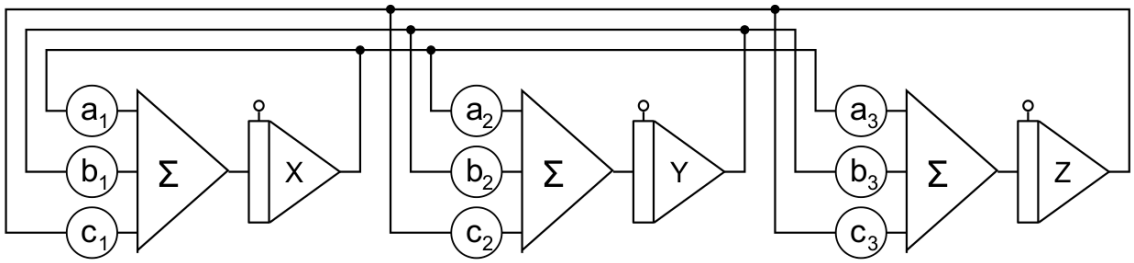
$$DYp_0 = \frac{h}{p} DY(p-1)_0 \tag{3.9}$$

Dostáváme algoritmus numerického výpočtu metodou Taylorovy řady. Výpočet se provede tak, že ze zadané počáteční podmínky vypočteme člen $DY1_0$. Z $DY1_0$ se dále stanoví člen $DY2_0$. Postup pokračuje až do p-tého členu. Nová hodnota řešení y_1 se určí jako součet počáteční podmínky a členů $DY1_0$ až DYp_0.

Uvedený postup lze aplikovat i na soustavu rovnic:

$$\begin{aligned} x' &= a_1x + b_1y + c_1z, & x(0) = x_0 \\ y' &= a_2x + b_2y + c_2z, & y(0) = y_0 \\ z' &= a_3x + b_3y + c_3z, & z(0) = z_0 \end{aligned} \tag{3.10}$$

Odpovídající schéma je na obrázku 3.2. Z blokového schéma je vidět možnost paralelizace a spolupráce procesorů. Každý integrátor se sčítačkou a násobičkami je jeden mikroprocesor.



Obrázek 3.2: Blokové schéma pro soustavu rovnic 3.10, Zdroj: [12]

Provedeme-li substituci soustavy rovnic 3.10, dostáváme:

$$\begin{aligned} x_1 &= x_0 + DX1_0 + DX2_0 \\ y_1 &= y_0 + DY1_0 + DY2_0 \\ z_1 &= z_0 + DZ1_0 + DZ2_0 \end{aligned} \tag{3.11}$$

kde význam jednotlivých členů je:

$$DX1_0 = h(a_1x_0 + b_1y_0 + c_1z_0) \tag{3.12}$$

$$DY1_0 = h(a_2x_0 + b_2y_0 + c_2z_0) \tag{3.13}$$

$$DZ1_0 = h(a_3x_0 + b_3y_0 + c_3z_0) \tag{3.14}$$

$$DX2_0 = \frac{h}{2}(a_1DX1_0 + b_1DY1_0 + c_1DZ1_0) \tag{3.15}$$

$$DY2_0 = \frac{h}{2}(a_2DX1_0 + b_2DY1_0 + c_2DZ1_0) \tag{3.16}$$

$$DZ2_0 = \frac{h}{2}(a_3DX1_0 + b_3DY1_0 + c_3DZ1_0) \tag{3.17}$$

Jak již bylo uvedeno, výpočet lze provést paralelně. Budou potřeba tři mikroprocesory. Posloupnost operací při řešení soustavy rovnic metodou Taylorovy řady 2. řádu je uvedena v tabulce 3.1.

Tabulka 3.1: Posloupnost operací při řešení soustavy 3.10

Číslo operace	**Mikroprocesor 1**	**Mikroprocesor 2**	**Mikroprocesor 3**
1	$X1 = a_1 \cdot x_0$	$Y1 = a_2 \cdot x_0$	$Z1 = a_3 \cdot x_0$
2	$X2 = b_1 \cdot y_0$	$Y2 = b_2 \cdot y_0$	$Z2 = b_3 \cdot y_0$
3	$X3 = c_1 \cdot z_0$	$Y3 = c_2 \cdot z_0$	$Z3 = c_3 \cdot z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = h \cdot X5$	$Y6 = h \cdot Y5$	$Z6 = h \cdot Z5$
7	$X7 = a_1 \cdot X6$	$Y7 = a_2 \cdot X6$	$Z7 = a_3 \cdot X6$
8	$X8 = b_1 \cdot Z6$	$Y8 = b_2 \cdot Y6$	$Z8 = b_3 \cdot Z6$
9	$X9 = c_1 \cdot Y6$	$Y9 = c_2 \cdot Y6$	$Z9 = c_3 \cdot Z6$
10	$X10 = X7 + X8$	$Y10 = Y7 + Y8$	$Z10 = Z7 + Z8$
11	$X11 = X10 + X9$	$Y11 = Y10 + Y9$	$Z11 = Z10 + Z9$
12	$X12 = \frac{h}{2}X11$	$Y12 = \frac{h}{2}Y11$	$Z12 = \frac{h}{2}Z11$
13	$X13 = X12 + X6$	$Y13 = Y12 + Y6$	$Z13 = Z12 + Z6$
14	$X14 = x_0 + X13$	$Y14 = y_0 + Y13$	$Z14 = z_0 + Z13$

Zdroj: Vlastní

3.3 Tvořící diferenciální rovnice

Velice často můžeme pozorovat, že funkce na pravé straně rovnic jsou určitého typu, často se vyskytujících v technické praxi (např. sin, cos, exp, atd). Funkce pomocí substitucí převedeme na nový systém rovnic, kde se vyskytují pouze polynomy na pravé straně. Nově vzniklým rovnicím říkáme tvořící diferenciální rovnice, protože tvoří původní funkci.

Například rovnice

$$y' = y + sin(t) \qquad y(0) = y_0 \tag{3.18}$$

se postupně převede na soustavu rovnic. Zaměříme se pouze na $sin(t)$

$$z = sin(t) \tag{3.19}$$

Zderivujeme a dostáváme

$$z' = cos(t) \tag{3.20}$$
$$z' = u \tag{3.21}$$

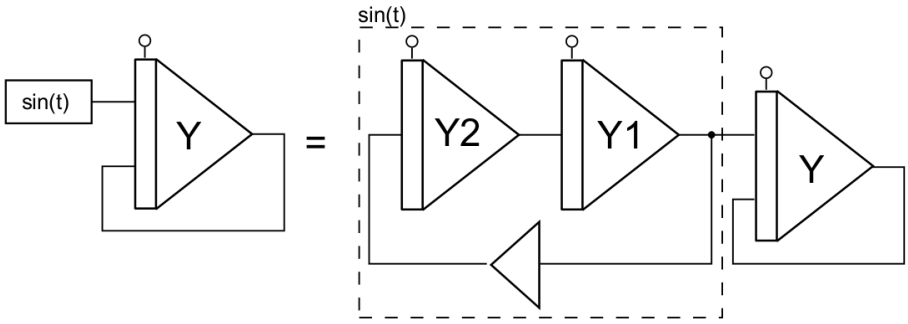
Znovu zderivujeme, tentokrát u

$$u = cos(t) \tag{3.22}$$
$$u' = -sin(t) \tag{3.23}$$

Dáme-li výsledky dohromady, tak máme výslednou soustavu rovnic

$$\begin{aligned} y' &= y + y_1 & y(0) &= y_0 \\ y_1' &= y_2 & y_1(0) &= 0 \\ y_2' &= -y_1 & y_2(0) &= 1 \end{aligned} \tag{3.24}$$

Počáteční podmínka y_1 je nastavena na 0, protože $sin(0) = 0$. Obdobně je počáteční podmínka y_2 nastavena na 1, protože $cos(0) = 1$. Nově jsou potřeba tři integrátory. Blokové schéma rovnice 3.18 před a po transformaci je na obrázku 3.3.



Obrázek 3.3: Blokové schéma rovnice 3.18 před a po transformaci, Zdroj: [16]

Kapitola 4

Specializovaný paralelní systém

Jedná se o paralelní systém vyvíjený na VUT FIT. Jeho autorem je Ing. Michal Kraus. Systém je určen pro řešení soustav diferenciálních rovnic Taylorovou řadou. Celý specializovaný paralelní systém je sestaven z:

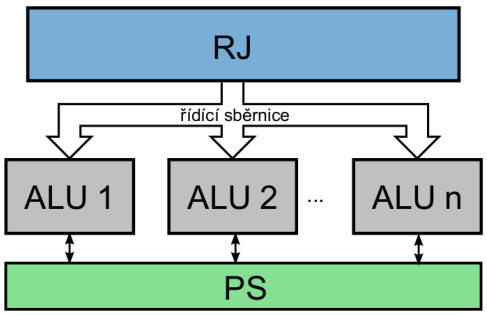
aritmeticko-logických jednotek ALU1, ALU2, ... ALUn provádějící operace odpovídající integrátorům, sčítačkám, odčítačkám, násobičkám a děličkám. Problematický je návrh integrátorů.

propojovacího systému PS realizující propojení vstupů a výstupů mezi jednotlivými jednotkami. Obtížné je efektivní propojení paralelního systému se stovkami až tisíci výpočetních uzlů.

řídící jednotky RJ generující posloupnost řídících signálů pro aritmeticko-logické jednotky.

řídící sběrnice propojující řídící jednotku RJ a aritmeticko-logické jednotky.

Blokové schéma systému je na obrázku 4.1.



Obrázek 4.1: Blokové schéma specializovaného paralelního systému, Zdroj: [15]

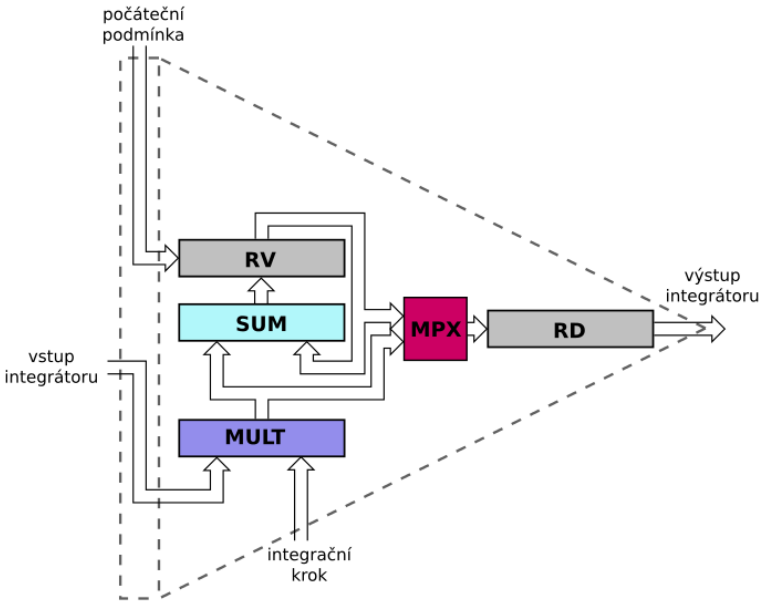
4.1 Numerické integrátory

Při numerickém výpočtu diferenciálních rovnic Taylorovou řadou se používají operace násobení a sčítání. Operace mohou být prováděny sériově nebo paralelně. Podobně komunikace mezi integrátory může probíhat sériově nebo paralelně. Proto integrátory rozdělujeme na

- paralelně-paralelní (P-P) integrátory = paralelní komunikace a paralelní výpočet,
- sériově-paralelní (S-P) integrátory = sériová komunikace a paralelní výpočet,
- sériově-sériové (S-S) integrátory = sériová komunikace a sériový výpočet.

4.1.1 Paralelně-paralelní (P-P) integrátor

Výpočet i komunikace probíhá paralelně. Výpočet realizuje paralelní sčítačka a paralelní násobička. Vstup a výstup je tvořen paralelní datovou sběrnicí.



Obrázek 4.2: Blokové schéma paralelně-paralelního (P-P) integrátoru, Zdroj:[11]

Bloky mají následující význam:

RV	registr výsledku
RD	registr součinu
MPX	multiplexor
SUM	paralelní sčítačka
MULT	paralelní násobička

Integrátor pracuje v několika krocích. Při zahájení výpočtu se do registru RD a RV uloží hodnota počáteční podmínky y_i. Až se na vstupu objeví hodnota $f(y_i)$, tak se nastaví integrační krok na h. Součin vstupu integrátoru a integračního kroku se vypočte v jednotce MULT a zapíše do registru RD a současně přičte k obsahu registru RV pomocí bloku SUM. V RD je hodnota $DY1$ a v RV je $y_i + DY1$. V dalším kroku se na vstupu objeví hodnota $f(DY1)$ a integrační krok $\frac{h}{2}$. Výpočet se opakuje dokud není dosaženo požadované přesnosti nebo maximálního počtu iterací.

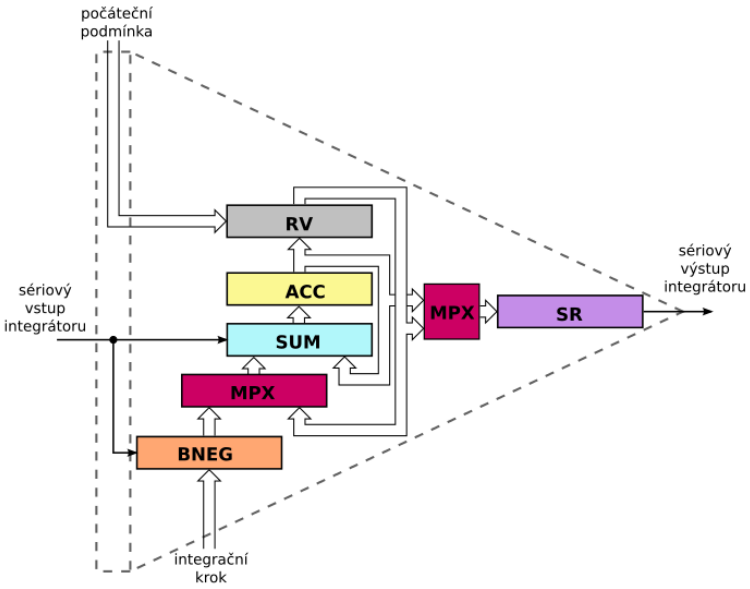
Čas na jeden krok výpočtu je dán zpožděním násobičky, sčítačky a zpožděním propojovací sítě.

$$t_{PP} = \tau_{nas} + \tau_{sec} + \tau_{sit} \tag{4.1}$$

Jedná se o nejrychlejší typ integrátoru. Nevýhodou je složitost zapojení, zejména paralelní násobičky. A dále šířka vstupní a výstupní sběrnice.

4.1.2 Sériově-paralelní (S-P) integrátor

Komunikace probíhá sériově, bit po bitu. Násobička pracuje také sériově. Je založena na principu Boothova algoritmu násobení (viz [13]), kdy se podle dvou sousedních bitů násobitele rozhoduje o průběhu výpočtu. Sčítačka je i nadále paralelní.



Obrázek 4.3: Blokové schéma sériově-paralelního (S-P) integrátoru, Zdroj: [14]

Bloky mají následující význam:

RV	registr výsledku
SR	posuvný registr
MPX	multiplexor
ACC	akumulátor
SUM	paralelní sčítačka
BNEG	obvod řízené negace pro sekvenční násobičku

Při zahájení výpočtu se do registru RV a SR uloží hodnota počáteční podmínky y_i. Integrační krok nastaven na h. Spodní multiplexor přepnut na cestu z BNEG. Dále se vynuluje akumulátor ACC. Na vstupu je připraven nejméně významový bit $f(y_i)$. Vstup integrátoru je násobitelem. Násobencem je integrační krok h. Postupně se načítají bity násobitele $f(y_i)$. Podle současného a předchozího bitu se rozhoduje, zda

- přičíst nulu k mezivýsledku,
- přičíst násobence (integrační krok h) k mezivýsledku,
- odečíst násobence (integrační krok h) od mezivýsledku, negace proběhne v bloku BNEG,

Výsledek ze SUM se zapíše do ACC, akumulátor a registr SR se posunou vpravo. Celý postup se opakuje, dokud není přijat a zpracován poslední bit ze vstupu $f(y_i)$. Tím dojde k sekvenčnímu vynásobení $f(y_i)$ a h. Výsledek násobení se uloží do registru SR. Multiplexor se přepne na cestu z registru RV, kde je uložen výsledek y_i předchozích iterací. Hodnota z RV se sečte s hodnotou DYp_i (výsledkem násobení vstupu a integračního kroku) v ACC a následně uloží do RV a SR.

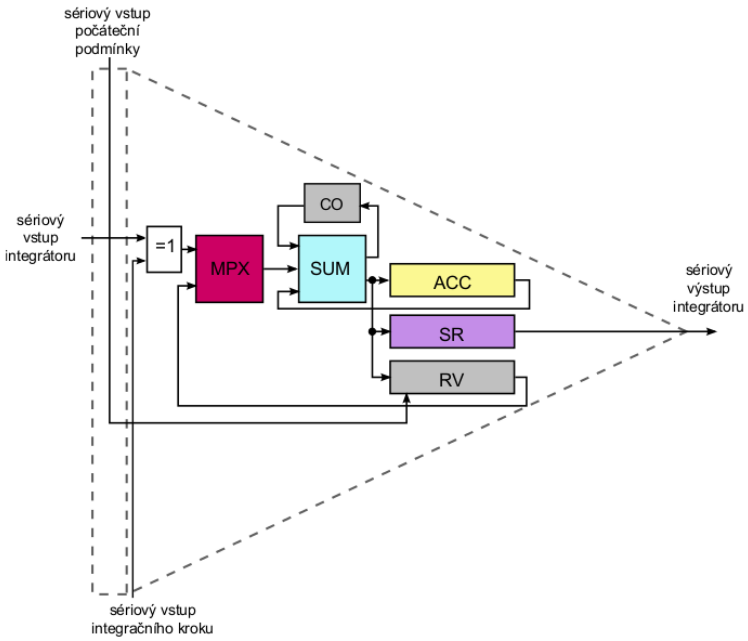
Výhodou S-P integrátoru je malý počet vývodů vstupního a výstupního rozhraní. Nevýhodou je zpomalení výpočtu, protože násobení se provádí v n krocích, kde n je počet bitů pro zobrazení čísel. Čas na jeden krok výpočtu je dán zpožděním násobičky (což je vlastně n kroků sčítání), zpožděním sčítačky a zpožděním propojovací sítě.

$$t_{SP} = \tau_{nas} + \tau_{sec} + \tau_{sit} \tag{4.2}$$

$$t_{SP} = n \cdot \tau_{sec} + \tau_{sec} + \tau_{sit} \tag{4.3}$$

4.1.3 Sériově-sériový (S-S) integrátor

Obě operace násobení a sčítání jsou realizovány sériově. Vychází z předchozí varianty S-P integrátoru. Komunikace je také sériová.



Obrázek 4.4: Blokové schéma sériově-sériového (S-S) integrátoru, Zdroj: [11]

Bloky mají následující význam:

RV	posuvný registr výsledku
SR	výstupní posuvný registr
ACC	akumulátor
SUM	jednobitová sčítačka
MPX	multiplexor
CO	klopný obvod pro uchování přenosu

Vynuluje se obsah ACC, obvod uchování přenosu CO a do registru RV se uloží hodnota počáteční podmínky. Multiplexor MPX se nastaví na cestu ze vstupu integrátoru. Na vstupu je připraven nejméně významový bit $f(y_i)$ a na vstupu integračního kroku je připraven nejméně významový bit integrační krok h. Podle bitu na vstupu integrátoru dojde k sečtení se vstupem integračního kroku do ACC. Celý postup se opět opakuje, až je v ACC hodnota DYp_i (výsledek násobení $f(y_i)$ a h). Pak se přepne multiplexor MPX na cestu z RV, kde je uložen výsledek y_i předchozích iterací. Obsahy registrů RV a ACC se postupně sečtou. Konečný výsledek se uloží do registru RV a SR.

S-S integrátor má nejmenší nároky na zapojení. Nevýhodou je rychlost výpočtu, která je daná vztahem

$$t_{SS} = \tau_{nas} + \tau_{sec} + \tau_{sit} \tag{4.4}$$

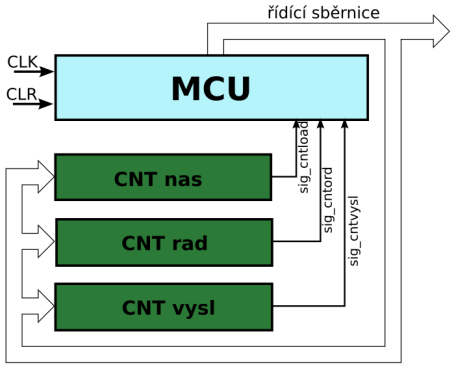
$$t_{SS} = n \cdot n \cdot \tau_{sec} + n \cdot \tau_{sec} + \tau_{sit} \tag{4.5}$$

$$t_{SS} = n^2 \cdot \tau_{sec} + n \cdot \tau_{sec} + \tau_{sit} \tag{4.6}$$

4.2 Řídící jednotka

Řídící jednotka se stará o řízení celého paralelního systému. Mezi její základní úkoly patří

- řízení výpočtu paralelního systému,
- zápis dat do všech výpočetních jednotek (integrátorů),
- distribuce výsledků.



Obrázek 4.5: Blokové schéma řadiče, Zdroj: [11]

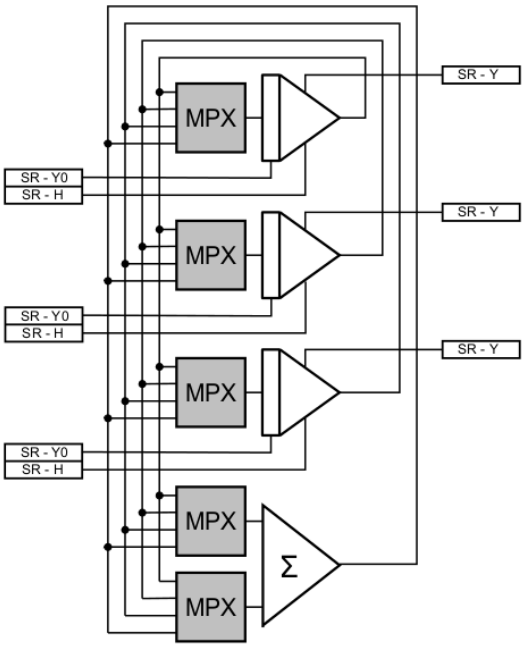
Řadič se příliš nepodobá univerzálním řadičům, protože je v paralelním systému pevně stanovena posloupnost řídících signálů. Je založen na principu mikroprogramového řadiče. Protože všechny integrátory provádějí stejný výpočet i když nad jinými daty, mohou být řízeny stejnou řídící jednotkou. Jedná se o architekturu typu **SIMD** (Single Instruction Multiple Data).

Důležitou části specializovaného řadiče je počítání smyček. Využívá se čítačů, které přímo ovládá řadič. Význam jednotlivých čítačů v blokovém schéma na obrázku 4.5 je následující:

- **CNT nas** = počítá počet součtů při násobení u S-P integrátoru nebo počet součtů sčítačky u S-S integrátoru,
- **CNT rad** = počítá řád Taylorovy řady DYp_i. Lze doplnit komparátorem, který porovnává dva poslední výsledky a v případě dosažení požadované přesnosti ukončí výpočet.
- **CNT vysl** = počítá počet výsledků y_i.

4.3 Propojovací síť

Propojovací síť musí zajistit propojení jednotlivých aritmeticko-logických jednotek (integrátorů, sčítaček, násobiček, ...) mezi sebou podle řešené diferenciální rovnice. Dále slouží k nahrání počáteční podmínky a integračního kroku a po dokončení výpočtu k nahraní výsledku. Posledním úkolem je napojení všech těchto jednotek sběrnicí na řídící systém.



Obrázek 4.6: Propojovací síť se třemi integrátory a jednou sčítačkou, Zdroj: [11]

Na obrázku 4.6 je ukázka sítě propojující tři integrátory a jednu sčítačku. Výpočetní jednotky jsou propojeny křížovými přepínači realizovanými pomocí multiplexorů. Jednotky jsou propojeny sériově z důvodu šetření plochy na čipu a vyšší přenosové rychlosti. Používají se tedy sério-paralelní integrátory. Každý multiplexor obsahuje registr s adresou, který vybírá, jaký vstup se přepne na výstup. Výstupy multiplexorů jsou připojeny na vstupy výpočetních jednotek.

K nahrání počátečních podmínek a integračních kroků se používá sériová sběrnice. Data jsou uložena v posuvných registrech SR-Y0 a SR-H a postupně se nahrávají do integrátorů. Po dokončení výpočtu se výsledky opět přes sériovou sběrnici nahrají do registrů SR-Y.

Kapitola 5

Simulace obvodů

Simulace obvodů je technika pro kontrolu a ověření správnosti návrhu elektrických a elektronických obvodů a systémů před jejich výrobou a nasazením. Používá se v širokém spektru aplikací, od integrovaných obvodů a mikroelektroniky po elektrické distribuční sítě. Simulace obvodů se skládá z

1. matematického modelu součástek obvodu (rezistory, cívky, atd.),
2. sestavení rovnic popisujících obvod,
3. technik pro řešení rovnic obvodu (viz kapitola 2).

Techniky pro simulaci obvodů byly poprvé představeny v 50. a 60. letech minulého století. V roce 1973 byl uveden L. W. Nagalem z Berkeley univerzity simulátor SPICE. [20]

5.1 Součástky

Uvažujme pouze lineární dvojbrany. Tedy rezistory, kondenzátory, cívky a zdroje.

5.1.1 Pasivní součástky

Rozlišujeme tři druhy pasivních součástek: rezistory, kondenzátory a cívky. Jejich charakteristika je následující:

Rezistory brání toku elektrického proudu dle Ohmova zákona $I = \frac{U}{R}$, kde R je elektrický odpor.

Kondenzátory uchovávají náboj ve svém elektrostatickém poli dle zákona $q = Cv$, kde q je elektrický náboj, C je hodnota kapacity. Závislost proudu na napětí, které je na kondenzátoru, lze vyjádřit vztahem $i = \frac{d}{dt}q = C\frac{dv}{dt}$.

Cívky uchovávají energii ve svém elektromagnetickém poli dle zákona $\Phi = Li$, kde Φ je magnetický tok, L je indukčnost. Závislost napětí na proud, který prochází cívkou, lze vyjádřit vztahem $v = \frac{d}{dt}\Phi = L\frac{di}{dt}$.

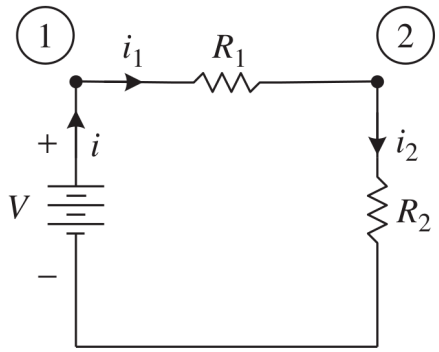
5.1.2 Nezávislé zdroje

Napěťové zdroje Ideální napěťový zdroj má na svých svorkách dané napětí, nezávisle na odebíraném proudu.

Proudové zdroje Ideální proudový zdroj dodává daný proud, nezávisle na tom, jaké napětí k tomu musí vyvinout. V praxi se s proudovými zdroji setkáme spíše jen výjimečně.

5.2 Sestavení rovnic

Chování obvodu je popsáno soustavou rovnic, která je formulována s využitím vzorců pro součástky a pomocí Kirchhoffových zákonů. Obecně je výsledkem soustava diferenciálních rovnic 1. řádu. Pro obvody obsahující pouze rezistory je výsledkem soustava algebraických rovnic.



Obrázek 5.1: Jednoduchý lineární obvod, Zdroj: [20]

Jako příklad uvažujme jednoduchý lineární obvod na obrázku 5.1. Z I. Kirchhoffova zákona vyplývá:

$$i = i_1 = i_2 \tag{5.1}$$

Ze vzorců pro součástky získáme rovnice:

$$u_1 = V, i_1 = \frac{u_1 - u_2}{R_1}, i_2 = \frac{u_2}{R_2} \tag{5.2}$$

Z I. Kirchhoffova zákona ($i_1 = i_2$) a II. Kirchhoffova zákona:

$$u_1 = R_1 i_1 + R_2 i_2 = (R_1 + R_2) i = (R_1 + R_2) \frac{u_2}{R_2} \tag{5.3}$$

Následně dostáváme:

$$u_2 = \frac{R_2}{R_1 + R_2} u_1 \tag{5.4}$$

Tento přístup k sestavení rovnic je špatně algoritmizovatelný a není vhodný pro velké obvody. Potřebujeme systematický a automatizovatelný postup:

Sparse Tableau Analysis (STA) Rovnice explicitně obsahují všechny proměnné obvodu jako napětí v uzlech, napětí na zdrojích napětí, proud tekoucí proudovými zdroji. I pro malé obvody se vygeneruje velké množství převážně krátkých rovnic.

Nodal Analysis (NA) Oproti STA cílem je vytvořit rovnice co nejjednodušší. Vektor neznámých obsahuje pouze napětí v uzlech. Proudové zdroje se těžko implementují. Cívky mohou vést na integro-diferenciální rovnice. Proto tato metoda není vhodná pro obvody, které obsahují proudové zdroje a cívky.

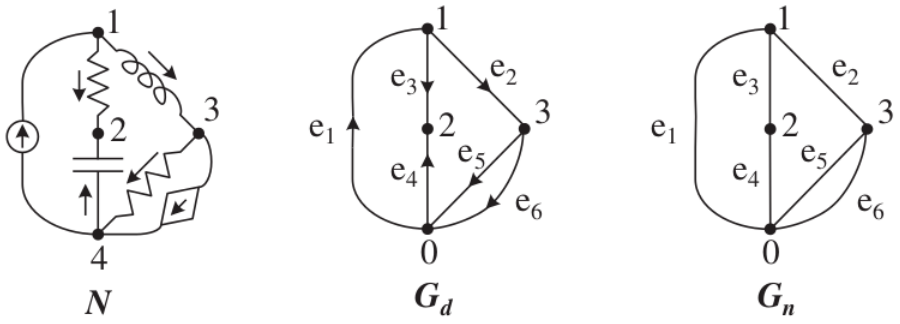
Modified Nodal Analysis (MNA) Je kompromisem mezi STA a NA. Kombinuje výhodné vlastnosti obou metod. Univerzálnost STA a menší množství rovnic u NA. Jedná se o nejvíce používanou metodu pro generovaní rovnic.

5.3 Reprezentace obvodu

Předtím než se budeme věnovat metodě MNA, je nutné vysvětlit, jak tyto metody reprezentují obvod. Vhodnou a přirozenou reprezentací obvodu je *orientovaný graf* G_d, kde

- vrcholy grafu odpovídají uzlům v obvodu a
- hrany grafu odpovídají jednotlivým součástkám, nikoliv větvím.

Orientace hrany koresponduje s technickým směrem proudu a napětí přes součástku (šipka ukazuje od + k −). Nezajímá-li nás orientace proudů a napětí v obvodu, můžeme použít *neorientovaný graf* G_n. Situaci ilustruje obrázek 5.2.



Obrázek 5.2: Obvod, jeho orientovaný a neorientovaný graf, Zdroj: [20]

5.4 Modifikovaná metoda uzlových napětí

Modifikovaná metoda uzlových napětí (anglicky *Modified Nodal Analysis*, dále označená jako **MNA**) byla představena Chung-Wen Ho roku 1975 v článku „The Modified Nodal Approach to Network Analysis“ ([7]). Je kompromisem mezi STA a NA. Kombinuje výhodné vlastnosti obou metod. Univerzálnost STA a menší množství rovnic u NA. Neznámými jsou

pouze napětí v uzlech e, proudy i_V tekoucí přes napěťové zdroje a proudy i_L tekoucí přes cívky. Tím je počet neznámých výrazně zredukován. Systém rovnic MNA je definován jako

$$A_C \frac{d}{dt} C A'_C e + A_R G A'_R e + A_L i_L + A_V i_v = -A_I I \tag{5.5}$$

$$\frac{d}{dt} L i_L - A'_L e = 0 \tag{5.6}$$

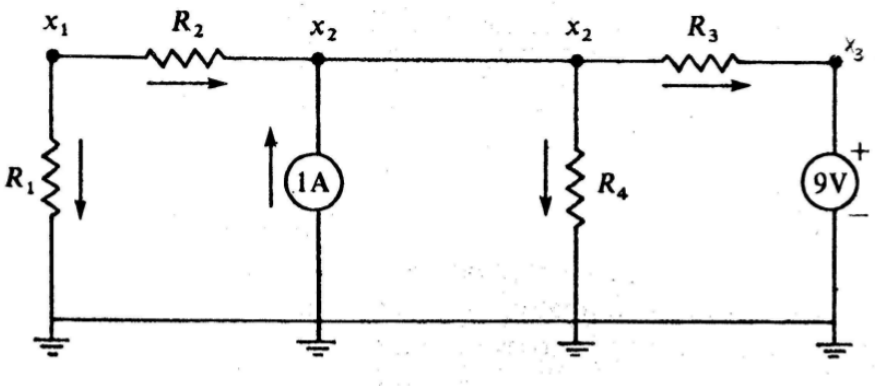
$$A'_V e = E \tag{5.7}$$

kde	A_R, A_C, A_L, A_V, A_I	jsou incidenční matice pro rezistory, kondenzátory, cívky, napěťové a proudové zdroje,
	e	je vektor napětí v jednotlivých uzlech,
	i_V	je vektor proudů tekoucích přes napěťové zdroje,
	i_L	je vektor proudů tekoucích přes cívky,
	I	je vektor hodnot proudových zdrojů,
	E	je vektor hodnot napěťových zdrojů,
	C	je diagonální matice obsahující kapacity všech kondenzátorů,
	G	je diagonální matice obsahující konduktivity všech rezistorů,
	L	je diagonální matice obsahující indukčnosti všech cívek.

Zápis rovnic v maticové formě

$$\begin{bmatrix} A_C C A'_C & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{d}{dt} \begin{bmatrix} e \\ i_L \\ i_V \end{bmatrix} + \begin{bmatrix} A_R G A'_R & A_L & A_V \\ -A'_L & 0 & 0 \\ A'_V & 0 & 0 \end{bmatrix} \begin{bmatrix} e \\ i_L \\ i_V \end{bmatrix} = \begin{bmatrix} -A_I I \\ 0 \\ E \end{bmatrix} \tag{5.8}$$

5.4.1 Ukázkový příklad



Obrázek 5.3: Obvod s rezistory, Zdroj: [20]

Na obrázku 5.3 je obvod obsahující pouze rezistory, napěťové a proudové zdroje. Tím se systém MNA rovnic 5.8 redukuje na

$$\begin{bmatrix} A_R G A'_R & A_V \\ A'_V & 0 \end{bmatrix} \begin{bmatrix} e \\ i_V \end{bmatrix} = \begin{bmatrix} -A_I I \\ E \end{bmatrix} \tag{5.9}$$

Vytvoříme incidenční matice.

$$A_R = \begin{bmatrix} +1 & +1 & 0 & 0 \\ 0 & -1 & +1 & +1 \\ 0 & 0 & -1 & 0 \end{bmatrix}, A_V = \begin{bmatrix} 0 \\ 0 \\ +1 \end{bmatrix}, A_I = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{5.10}$$

Dále diagonální matici G obsahující konduktivity všech rezistorů, vektor I pro proudové zdroje a vektor E pro napěťové.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, I = \begin{bmatrix} 1 \end{bmatrix}, E = \begin{bmatrix} 9 \end{bmatrix} \tag{5.11}$$

Dostáváme konečný systém rovnic v maticové formě, který můžeme vyřešit např. v Matlabu.

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 2 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ i_V \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 9 \end{bmatrix} \tag{5.12}$$

5.4.2 Omezení metody

Jak je vidět z rovnice 5.8, proměnnými nejsou všechny proudy v obvodu. Problémem metody je určení všech proudů. Máme dvě možnosti:

1. Proudy dopočítat. Například známe-li úbytek napětí na libovolném rezistoru a jeho odpor, tak proud vypočítáme z Ohmova zákona.

2. Využít triku se zdrojem napětí. Zajímá-li nás opět proud rezistorem, tak k němu do série zapojíme zdroj napětí s hodnotou $0V$. Tedy vliv zdroje na obvod bude nulový, ale z MNA získáme proud, který jím teče. Tento proud je roven proudu, který teče zkoumaným rezistorem.

Kapitola 6

Návrh

Cílem práce je navrhnout aplikaci vykreslující frekvenční charakteristiky k zadanému obvodu. V této kapitole se nejprve budeme zabývat požadavky na aplikaci a poté samotným návrhem s popisem jednotlivých modulů.

6.1 Specifikace požadavků

V této podkapitole budou stanoveny všechny požadavky na výslednou aplikaci.

Nezávislost na architektuře Výsledná aplikace musí být přenositelná. Minimálně funkční na operačních systémech Linux a Windows.

Jednoduché uživatelské rozhraní Uživatel by se měl během krátké chvíle zorientovat v aplikaci a dále s ní pracovat bez nutnosti čtení manuálu.

Modularita Aplikace by se měla skládat z modulů (tříd) s daným rozhraním. Pak by neměl být problém použít jiný modul např. pro vykreslení grafu nebo analýzu obvodu.

Pasivní kmitočtové filtry Vzhledem k tomu, že výstupem programu budou frekvenční charakteristiky, tak modelovanými obvody budou kmitočtové filtry. Konkrétně obvody skládající se pouze z pasivních prvků, tedy z

- rezistorů,
- kondenzátorů a
- cívek.

Vyznačují se tím, že jsou frekvenčně závislé. Velikost amplitudy a fázového posunu výstupního napětí je závislá na frekvenci vstupního střídavého napětí.

Diferenciální rovnice Vhodné pro matematický popis chování obvodu. Existují i další možnosti jako komplexní čísla nebo v případě kmitočtových filtrů tzv. přenosová funkce.

Zobrazení sestavených diferenciální rovnic Jedním ze zamýšlených účelů aplikace je její využití ve výuce. Proto je vhodné, aby si uživatel mohl pro zapojený obvod zobrazit použité diferenciální rovnice.

TKSL/C Systém určený pro rychlé a přesné řešení diferenciálních rovnic. V navrženém programu bude použit právě tento systém pro řešení sestavených rovnic. V budoucnu bude možné napojit na TKSL specializovaný paralelní systém a výpočet akcelerovat na FPGA.

Frekvenční charakteristiky Budou výstupem programu. Frekvenční charakteristiky jsou souhrnným názvem pro amplitudovou a fázovou charakteristiku.

- Amplitudová charakteristika = zachycuje závislost přenosu, což je podíl výstupního a vstupního napětí, na frekvenci.
- Fázová charakteristika = zachycuje závislost fázového posunu mezi výstupním a vstupním napětím na frekvenci.

Požadavkem na aplikaci je možnost vykreslit průběhy z více simulací do jednoho grafu a možnost přepínat mezi výsledky.

Koeficienty zrychlení Koeficienty zrychlení říkají, jak se změní doba výpočtu při použití různých verzí integrátorů ve specializovaném paralelním systému a také jak by se změnila v případě, že bychom neměli k dispozici dostatečné množství výpočetních jednotek. V programu musí být vypočítány koeficienty pro různé počty dostupných výpočetních jednotek.

Nastavení hodnot Uživatel musí mít možnost nastavit hodnoty všech součástek a rozsah měření pro simulaci. Všechny hodnoty by mělo jít změnit v dialogu před zahájením simulace. Hodnoty součástek musí jít změnit i dvojklikem na danou součástku.

Osciloskopy Pro měření výstupu se bude muset připojit do obvodu osciloskop. Osciloskopů bude moci být v obvodu více. Použit bude ten osciloskop, který uživatel vybere v dialogu při zahájení simulace.

Souhrnné informace o simulaci Pro každý běh simulace se musí uložit všechny potřebné hodnoty. Uživatel si je bude moci zpětně procházet a porovnávat. Důležitými informacemi o simulaci jsou

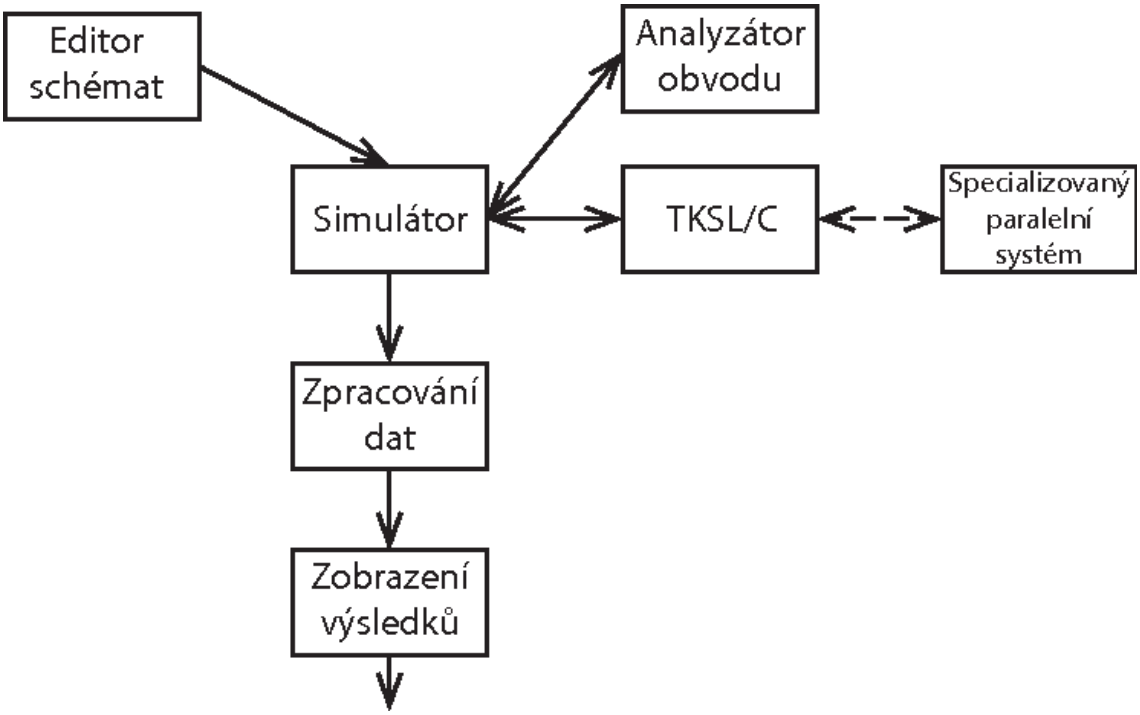
- seznam použitých součástek a jejich hodnoty,
- rozsah proměřených frekvencí,
- použitý osciloskop pro měření výstupu,
- sestavené diferenciální rovnice a
- vypočítané koeficienty zrychlení.

Historie spuštěných simulací Simulace bude možné spouštět opakovaně a dokonce i měnit schéma zapojeného obvodu. Historie se všemi údaji o simulaci musí být uživateli zpětně přístupná.

6.2 Návrh programu

Navržený program obsahuje grafické uživatelské rozhraní, nejedná se o konzolovou aplikaci. Činnost programu by bylo možné rozdělit do dvou základních bloků. Prvním je editor elektrických schémat. Druhým blokem je analyzátor obvodu a vlastní simulace.

Zjednodušené schéma programu je na obrázku 6.1. Uživatel nejprve nakreslí schéma obvodu v editoru, pak spustí simulaci. V simulaci se provede analýza obvodu, jejíž výstupem jsou sestavené diferenciální rovnice. Následně je tato soustava rovnic vyřešena v systému TKSL/C. Původně měl být výpočet TKSL/C ještě akcelerován ve specializovaném paralelním systému na čipu FPGA (Fitkit). Bohužel v současné době chybí rozhraní mezi TKSL/C a paralelním systémem. Řešení rovnic v TKSL/C musí probíhat opakovaně, vždy s jinou frekvencí tak, aby se proměřil celý rozsah frekvencí požadovaný uživatelem. V rámci každé dekády se proměří 10 hodnot. V každém kroku se po vyřešení rovnic v modulu zpracovávající výsledky odečtou potřebné hodnoty. Vypočte se přenos filtru a fázový posun. Na konci celé simulace se výsledky předají do grafu, který je zobrazí uživateli. Zároveň se uloží všechna potřebná data o právě skončené simulaci tak, aby byla později přístupná uživateli.



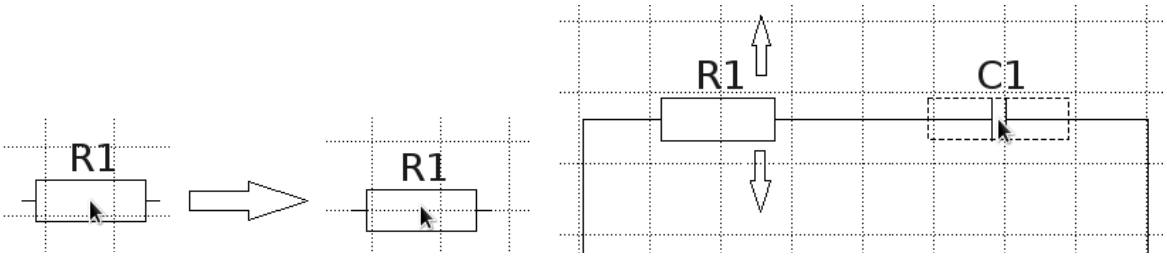
Obrázek 6.1: Návrh programu, Zdroj: Vlastní

6.3 Modul pro editaci schémat

Modul navržený pro schématické zadávání vstupního elektrického obvodu. Inspirací při návrhu editoru z pohledu uživatelského rozhraní byly profesionální programy jako SPICE nebo Matlab. Editor je navržen tak, aby práce s ním byla uživatelsky přívětivá. Na pozadí bude mřížka, ke které se součástky v určité vzdálenosti od ní samy přichytí (obrázek 6.2 vlevo). Rovněž práce s vodiči a součástkami nebude vyžadovat příliš grafické přesnosti od uživatele. Kdy při kliknutí na součástku se podle její orientace (vodorovně nebo svisle) vodič automaticky přichytí na její okraj.

Se součástkami bude dále možné hýbat a otáčet s nimi. Pokud budou na ni připojeny vodiče, tak se budou pohybovat spolu se součástkou. Při pohybu vodičů se bude měnit

vhodně pozice i na nich připojených součástek nebo jiných vodičů. Například při pohybu se součástkou C_1 na obrázku 6.2 vpravo se bude zároveň pohybovat i součástka R_1.



Obrázek 6.2: Automatické přichycení součástky ke mřížce a pohyb se součástkou, Zdroj: Vlastní

6.4 Modul pro simulaci

Úkolem modulu bude spustit analyzátor obvodu a vyřešit sestavené rovnice v TKSL/C. Řešení rovnic bude spouštět v TKSL/C vždy s jinou vstupní frekvencí zdroje. Frekvence se budou nastavovat podle požadovaného rozsahu měření. Daný rozsah frekvencí vhodně proměří, v každé dekádě spustí simulaci desetkrát. Výsledné hodnoty v jednotlivých iteracích předá modulu, který z nich odečte potřebné hodnoty pro výpočet přenosu filtru a fázového posunu na dané frekvenci.

6.5 Modul analyzující obvod

Metoda analyzující obvod je popsána dále v podkapitole 7.2.1 a vychází z již představených metod jako STA nebo MNA, které jsou popsaných v podkapitole 5.2. Chování obvodu popíše pomocí diferenciálních rovnic.

Metoda nejprve očísluje uzly v obvodu, který mají stejný potenciál vzhledem k zemi. Pak sestaví pomocnou soustavu rovnic na základě:

- I. Kirchhoffova zákona pro proudy v uzlech,
- II. Kirchhoffova zákona pro napětí,
- rovnic součástek uvedených v podkapitole 5.1.1.

Z této pomocné soustavy rekurzivně sestaví výslednou soustavu rovnic. Rovnice budou dále optimalizovány. Výsledkem modulu bude soustava diferenciálních rovnic, kterou bude možné následně vyřešit v TKSL/C.

6.6 Modul pro zpracovaní výsledků simulace

Úkolem modulu je zpracovat výstup z programu TKSL/C a z těchto dat odečíst potřebné hodnoty k výpočtu přenosu a fázového posunu. Zajímá nás

- maximální hodnota vstupního napětí,

- maximální hodnota výstupního napětí a
- časy těchto maxim.

Ve skutečnosti bude stačit odečíst pouze hodnoty pro výstupní napětí. Maximu vstupního napětí odpovídá velikost napětí na vstupním zdroji (ten bude v obvodu pouze jeden). Čas tohoto maxima lze snadno vypočítat. Z těchto hodnot je podle vzorců uvedených v podkapitole 7.3.4 možné vypočítat přenos filtru a fázový posun.

6.7 Modul pro zobrazení výsledků

Modul vykreslí data do grafu a ten zobrazí uživateli. V grafu bude možné vykreslit více křivek a tak porovnávat výsledky z různých simulací. Graf bude pracovat ve dvou režimech, v režimu pro vykreslení amplitudové charakteristiky a v režimu pro vykreslení fázové charakteristiky. Součástí grafu bude legenda, dále bude moci uživatel měnit barvu křivek pro zvolenou simulaci.

Kapitola 7

Implementace

Tato kapitola se věnuje popisu implementace a problémům, na které jsem narazil. Prezentuje popis jejich řešení a případně navrhuje možná další vylepšení. Cílem kapitoly ovšem není jít do implementačních detailů, ty jsou součástí komentářů ve zdrojových kódech. Bližší popis výsledné aplikace a jejího ovládání je uveden v příloze B.3.

7.1 Použité nástroje

Program je napsán v jazyce C++ za použití frameworku Qt. C++ je objektově orientovaný programovací jazyk, jehož autorem je Bjarne Stroustrup. Qt je jednou z nejpopulárnějších multiplatformních knihoven pro vytváření programů s grafickým uživatelským rozhraním. Aplikace v Qt používají nativní vzhled OS pro grafické uživatelské prostředí, takže se aplikace vždy přizpůsobí do používaného prostředí. Qt je knihovna pro programovací jazyk C++, ale existuje i pro jiné jazyky jako Python (PyQt), Ruby (QtRuby) a jiné.

Qt používá například webový prohlížeč Opera, populární linuxové desktopové prostředí KDE, komunikátor Skype, virtualizační software VirtualBox a další aplikace. Qt běží i na nejrůznějších mobilních zařízeních. Podporuje nejen unixové operační systémy jako Linux a BSD, ale i Windows a Mac OS X. Celý framework je dostupný zdarma pod licencí GPL nebo za poplatek pro komerční použití.

Dalším nástrojem je systém TKSL/C (Taylor-Kunovský Simulation Language), ve kterém probíhá řešení sestavených rovnic. Systém provádí numerické řešení pomocí Taylorova rozvoje. Přesnost je udržována dynamickým nastavováním řádu Taylorova polynomu. Systém je napsán v programovacím jazyce C++ a je určen pro operační systémy Windows a Linux. TKSL/C je nová verze simulačního nástroje TKSL/386.

7.2 Analyzátor obvodu

Analyzátor obvodu sestaví diferenciální rovnice popisující daný obvod. Navržená metoda vychází z metod popsaných v kapitole 5.2. Problémem popisovaných metod je jejich řešení v TKSL respektive jejich možný převod do formátu zápisu rovnic v TKSL.

7.2.1 Použitá metoda pro sestavení rovnic

Jak již bylo uvedeno, princip použité metody vychází ze základů metod STA a MNA představených v kapitole 5. Společnou částí je sestavení rovnice pro I. a II. Kirchhoffův zákon a

rovnic pro součástky. Rozdíl spočívá ve způsobu, jakým se rovnice „spojí“ do jedné výsledné soustavy rovnic.

Metoda pracuje v několika krocích:

1. Očíslovat uzly v obvodu se stejným potenciálem

   Všechny uzly v obvodu musí být očíslovány. Uzel s číslem 0 je vybrán jako tzv. zem. Tímto uzlem je vždy jeden z uzlů zdroje napětí, jelikož v obvodu může být zapojen pouze jeden. Uzly se stejným potenciálem vzhledem k zemi mají stejné číslo.

2. Sestavit rovnice pro I. Kirchhoffův zákon

   Pro každý uzel v obvodu sestavíme rovnice proudů dle I. Kirchhoffova zákona.

3. Sestavit rovnice pro II. Kirchhoffův zákon

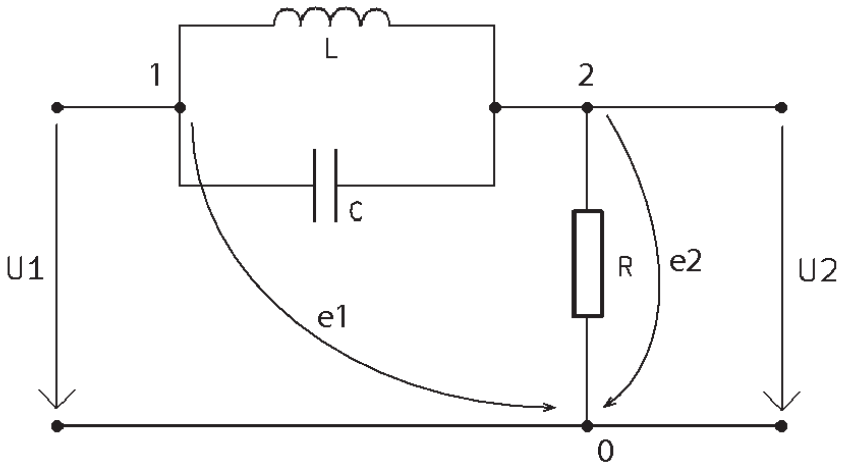
   Pro každou součástku vyjádříme úbytek napětí jako rozdíl napětí mezi jednotlivými uzly. Každý uzel má nějaký potenciál vzhledem k zemi.

4. Sestavit rovnice pro součástky

   Dle kapitoly 5.1.1 pro každou součástku sepíšeme její rovnici. Pro zdroje napětí se použije rovnice $u = sin\omega t$. Pro osciloskopy rovnice $i_{OSC} = 0$, protože měřícími přístroji nemůže protékat proud.

5. Sloučit rovnice do jedné soustavy rovnic

   Výsledná soustava rovnic musí být v zápisu vhodném pro TKSL. Metoda, která soustavu rovnic sestaví, pracuje rekurzivně. Začíná s rovnicemi součástek pro kondenzátory a cívky. Postupně odebírá rovnice a dosazuje je do výsledné soustavy. Může-li v jednom kroku použít více než jednu rovnici, tak postupně vyzkouší všechny možnosti. Pokud se nepodaří vyjádřit všechny potřebné proměnné, tak dojde k navracení a vyzkouší se jiná možnost. Výpočet končí, když se podaří všechny proměnné vyjádřit.



Obrázek 7.1: RLC obvod, Zdroj: Vlastní

Příklad

Příklad ilustruje použití metody. Mějme obvod na obrázku 7.1, který se chová jako filtr typu pásmová zádrž, s rezistorem, kondenzátorem a cívkou. Pro jednoduchost obvodu nebudeme uvažovat použití osciloskopů. Napětí U_1 představuje zdroj napětí. Jednotlivé uzly již máme očíslovány, proto můžeme začít sestavovat pomocné rovnice.

Rovnice pro I. KZ

$$-i_{U1} - i_L - i_C = 0 \tag{7.1}$$
$$i_L + i_C - i_R = 0 \tag{7.2}$$

Rovnice pro II. KZ

$$u_{U1} = e_1 \tag{7.3}$$
$$u_L = e_1 - e_2 \tag{7.4}$$
$$u_C = e_1 - e_2 \tag{7.5}$$
$$u_R = e_2 \tag{7.6}$$

Rovnice součástek

$$u_{U1} = sin\omega t \tag{7.7}$$
$$i'_L = \frac{1}{L}u_L \tag{7.8}$$
$$u'_C = \frac{1}{C}i_C \tag{7.9}$$
$$i_R = \frac{u_R}{R} \tag{7.10}$$

Postup sestavení výsledné soustavy rovnic je uveden v tabulce 7.1. Výsledná soustava rovnic obsahuje rovnice:

$$i'_L = \frac{1}{L}u_L \tag{7.11}$$
$$u'_C = \frac{1}{C}i_C \tag{7.12}$$
$$u_L = e_1 - e_2 \tag{7.13}$$
$$i_C = i_R - i_L \tag{7.14}$$
$$e_1 = u_{U1} \tag{7.15}$$
$$e_2 = e_1 - u_C \tag{7.16}$$
$$i_R = \frac{u_R}{R} \tag{7.17}$$
$$u_{U1} = sin\omega t \tag{7.18}$$
$$u_R = e_2 \tag{7.19}$$

Díky tomu, že metoda pracuje rekurzivně, tak se jí podaří pro každý platný obvod sestavit soustavu rovnic.

Tabulka 7.1: Posloupnost operací při sestavování výsledné soustavy rovnic

Krok	**Přidaná rovnice**	**Chybějící proměnné**	**Poznámka**
1.	$i'_L = \frac{1}{L}u_L$	u_L	
2.	$u'_C = \frac{1}{C}i_C$	u_L, i_C	
3.	$u_L = e_1 - e_2$	i_C, e_1, e_2	
4.	$i_C = i_R - i_L$	e_1, e_2, i_R	
5.	$e_1 = ?$	e_1, e_2, i_R	Lze použít 2 rovnice. Větev A a B.
5.A	$e_1 = u_C + e_2$	e_2, i_R	
6.A	$e_2 = u_R$	i_R, u_R	
7.A	$i_R = \frac{u_R}{R}$	u_R	
8.A	$u_R = ?$	u_R	Nelze použít žádnou rovnici. Navrácení!
5.B	$e_1 = u_{U1}$	e_2, i_R, u_{U1}	
6.B	$e_2 = ?$	e_2, i_R, u_{U1}	Lze použít 2 rovnice. Větev B.A a B.B.
6.B.A	$e_2 = u_R$	i_R, u_{U1}	
7.B.A	$i_R = \frac{u_R}{R}$	u_{U1}, u_R	
8.B.A	$u_{U1} = sin\omega t$	u_R	
9.B.A	$u_R = ?$	u_R	Nelze použít žádnou rovnici. Navrácení!
6.B.B	$e_2 = e_1 - u_C$	i_R, u_{U1}	
7.B.B	$i_R = \frac{u_R}{R}$	u_{U1}, u_R	
8.B.B	$u_{U1} = sin\omega t$	u_R	
9.B.B	$u_R = e_2$		Konec. Soustava rovnic sestavena.

Zdroj: Vlastní

7.2.2 Měření na osciloskopech

V obvodu může být zapojeno libovolné množství osciloskopů. Před spuštěním simulace je nutné v dialogovém okně zvolit ten osciloskop, na kterém se bude měřit výstupní napětí. Na vstup není potřeba připojovat osciloskop. Součástkami, které jsou připojeny ve větvi s osciloskopem, neteče žádný proud a je na nich nulový úbytek napětí.

7.2.3 Zkratované součástky

Výhodou použité metody je, že se nemusí se zkratovanými součástkami zacházet speciálním způsobem. Na takové součástce je nulový úbytek napětí a nemá žádný vliv na chování obvodu. Zkratovaná součástka má oba uzly ohodnoceny stejným číslem. Tedy rozdíl napětí mezi oběma uzly je nula.

7.3 Simulátor řešící sestavené rovnice

Úkolem modulu simulátoru je vyřešit rovnice v TKSL/C a zpracovat naměřené hodnoty. Z vlastností elektrických obvodů vyplývá problém s ustálením výstupního napětí. Z povahy modelovaní spojitých systémů na diskrétním počítači, plyne problém volby integračního kroku a lokalizace maxima.

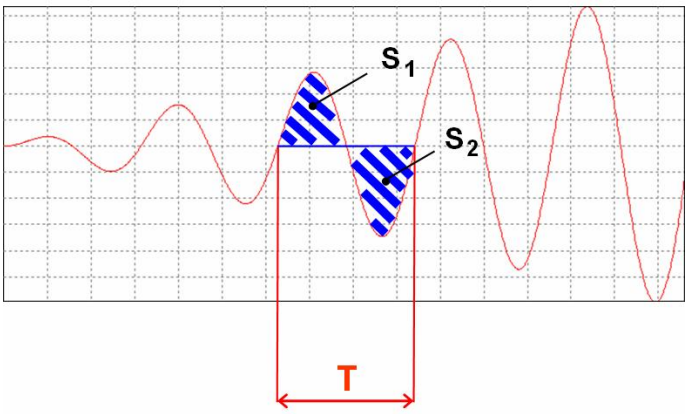
7.3.1 Napojení na TKSL/C

Program vytvoří nový proces, ve kterém spustí binární soubor s TKSL/C. Souboru předá všechny potřebné informace jako délka běhu, integrační krok a rovnice. TKSL/C vypisuje spočítané hodnoty na svůj výstup. Ve svém programu tyto hodnoty čtu a ukládám do paměti.

7.3.2 Ustálení výstupního napětí

Přesnost frekvenční charakteristiky ovlivňuje zejména přesnost odečtení amplitudy vstupního a výstupního napětí. Odečet hodnot je nutné provádět v místě, kde je výstupní napětí skutečně ustáleno. Rychlost ustálení je závislá na parametrech obvodu a frekvenci vstupního napětí. Tento problém ilustruje obrázek 7.2, kde můžeme vidět neustálený harmonický signál. Pro ustálený harmonický signál musí platit, že kladná i záporná půlvlna mají stejnou plochu, tj. $|S_1| = |S_2|$. V praxi je zbytečné počítat plochu půlvln, máme dvě možnosti

- porovnat hodnoty po sobě jdoucích amplitud v kladné půlvlně,
- porovnat hodnoty kladné a záporné amplitudy v rámci jedné periody.



Obrázek 7.2: Neustálený signál, Zdroj: [27]

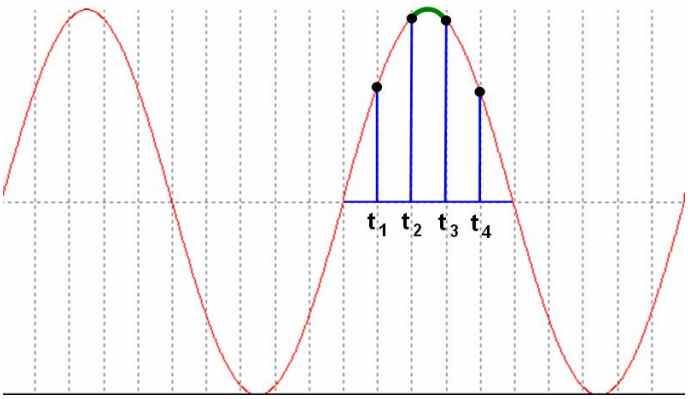
Je-li dosaženo požadované přesnosti, je možné považovat signál za ustálený. Čím delší bude běh simulace, tím je vyšší pravděpodobnost, že signál bude ustálený. Proto v programu neprovádím žádnou kontrolu a délka běhu simulace je pevně stanovena na 10 period vstupního signálu.

Pokud by bylo požadováno provádět tuto kontrolu, tak není nutné výpočet spouštět znovu. Stačí použít poslední hodnoty z minulého běhu a ty nastavit jako počáteční podmínky integrátorů. Tím bude výpočet plynule pokračovat a žádné hodnoty se nebudou počítat znovu. Musíme si ale uvědomit, že tak dojde k „posunu“ času a další simulace (na stejné frekvenci) ve skutečnosti nepoběží od nuly.

7.3.3 Volba integračního kroku a lokalizace amplitudy

Simulace na počítači s sebou přináší několik problémů. Problémy vyplývají ze samotné podstaty. Kdy se snažíme řešit na počítači, který pracuje diskrétně, spojitý systém jako je například elektrický obvod. [9]

S tím úzce souvisí volba integračního kroku. Pokud zvolíme krok příliš malý, simulace bude trvat zbytečně dlouho. Zvolíme-li integrační krok příliš velký, tak se dopustíme nepřesnosti (viz obrázek 7.3). Velikost integračního kroku nastavuji tak, aby se v každé periodě navzorkovalo minimálně 100 hodnot. TKSL si případně velikost integračního kroku zmenší. Ideálním řešením by bylo hrubě lokalizovat maximum a kolem této hodnoty spustit simulaci znovu s velice malým krokem.



Obrázek 7.3: Problém nalezení maxima, Zdroj: [27]

7.3.4 Zpracování dat z TKSL/C

Z důvodu efektivity a také proto, že nás zajímá velikost amplitudy v poslední periodě běhu simulace, čtu hodnoty odzadu. Z dat musím odečíst amplitudu pro výstupní napětí u_2 a čas t_2, kdy se tak stalo. Protože vstupní napětí má průběh jako sinusovka, tak nemusím tyto hodnoty odečítat, ale můžu si je vypočítat.

Následně vypočtu přenos. Přenos se vypočítá jako poměr výstupního a vstupního napětí. Pokud není přenos přepočítán na decibely, tak se jedná o bezrozměrnou veličinu, tedy

$$A_U = \frac{U_2}{U_1} \tag{7.20}$$

kde A_U je přenos filtru [-]
U_1 je vstupní napětí [V]
U_2 je výstupní napětí [V]

Někdy se přenos vyjadřuje v decibelech podle vztahu

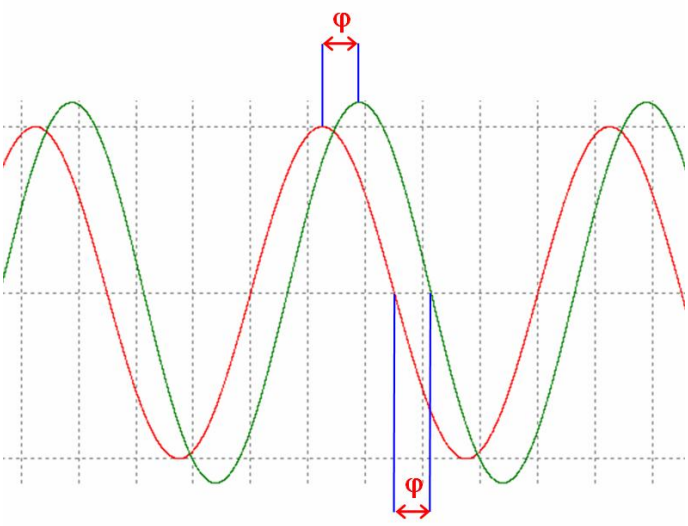
$$A_{UdB} = 20 \cdot log\left(\frac{U_2}{U_1}\right) \tag{7.21}$$

kde A_{UdB} je přenos filtru v decibelech [dB]
U_1 je vstupní napětí [V]
U_2 je výstupní napětí [V]

Posledním krokem je výpočet fázového posunu. K výpočtu fáze lze použít již dříve nalezené časy pro amplitudy nebo časy průchodů signálů nulou. Oba způsoby jsou ekvivalentní. Situaci ilustruje obrázek 7.4.

$$\varphi = \frac{-360}{T}\Delta t = -360 f(t_2 - t_1) \tag{7.22}$$

kde	T je perioda vstupního signálu	[s]
	f je frekvence vstupního signálu	[Hz]
	Δt je časový posun výstupního napětí	[s]
	t_1 je čas odečtení hodnoty maxima vstupního napětí	[s]
	t_2 je čas odečtení hodnoty maxima výstupního napětí	[s]



Obrázek 7.4: Možnosti odečtení fázového posunu, Zdroj: [27]

Předbíhá-li výstupní napětí vstupní, tak je fáze kladná, pokud je výstupní napětí zpožděno, tak je fáze záporná.

7.4 Koeficienty zrychlení

Koeficienty zrychlení udávají rychlost výpočtu rovnic při použití jednotlivých verzí integrátorů popsaných v kapitole 4. Podle provedeného srovnání (viz [11]) mezi různými typy integrátorů byla stanovena relativní rychlost mezi nimi. Srovnání je uvedeno v tabulce 7.2.

Tabulka 7.2: Srovnání rychlosti mezi jednotlivými typy integrátorů

Typ integrátoru	**Koeficient zrychlení integrátoru**
P-P	1
S-P	10
S-S	200

Zdroj: Vlastní

Tabulka 7.2 říká, že nejrychlejším typem integrátoru je paralelně-paralelní integrátor. Je přibližně desetkrát rychlejší než sériově-paralelní integrátor a dvěstěkrát rychlejší než sériově-sériový.

V tabulce 3.1 byla uvedena posloupnost operací při řešení blokového schéma se třemi integrátory. Všechny integrátory pracovaly paralelně, protože výpočet každého integrátoru byl řešen na jednom procesoru. Koeficienty zrychlení pro celé blokové schéma by v tomto případě odpovídaly hodnotám pro jednotlivé integrátory. Koeficienty zrychlení navíc zohledňují situaci, kdy počet potřebných integrátorů (procesorů) k řešení rovnice je menší než počet dostupných integrátorů (procesorů). To znamená, že výpočet bude trvat déle, minimálně dvojnásobnou dobu. Uvažujme případ, kdy řešíme rovnici se třemi integrátory, ale máme pouze dva procesory. Pak bude výpočet trvat dobu t, protože nebudeme moci každému integrátoru přiřadit jeden procesor a některý procesor bude muset být sdílený. Budeme-li mít tři procesory, tak výpočet bude probíhat paralelně a bude trvat dobu $\frac{t}{2}$. V navrženém programu jsou vždy uvedeny příslušné koeficienty zrychlení pro případy, kdy máme k dispozici jeden, dva, čtyři nebo osm procesorů.

Mezi celkovou dobou výpočtu a koeficientem zrychlení platí přímá úměra. Čím vyšší je koeficient zrychlení, tím je celková doba výpočtu delší. Známe-li celkovou dobu výpočtu například pro P-P integrátor, tak přibližnou dobu výpočtu při použití ostatních dvou typů integrátoru můžeme vypočítat vynásobením této doby výpočtu s příslušným koeficientem zrychlení integrátoru.

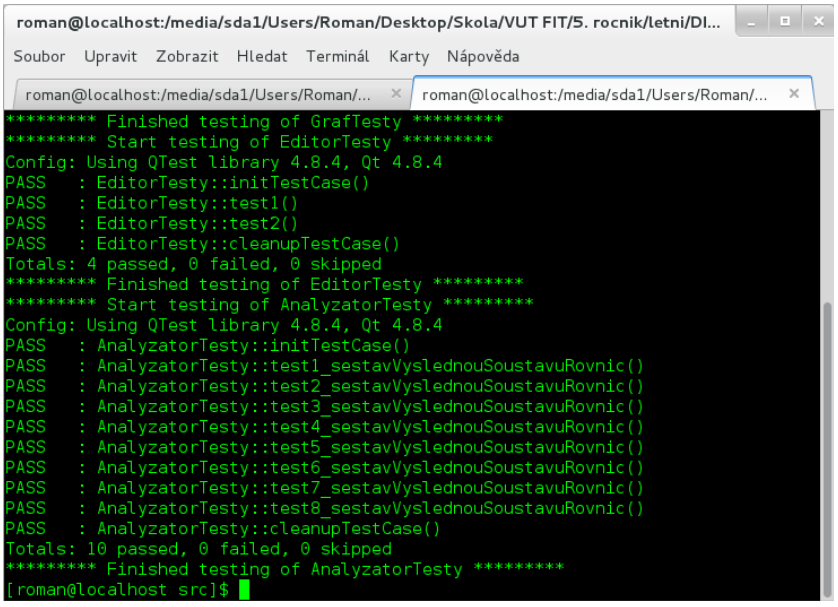
Kapitola 8

Testování

Testování je nedílnou součástí vývoje softwaru. Většina velkých firem vynakládá nemalé množství finančních prostředků na vývoj automatizovaných testovacích nástrojů. Účelem testování je nalezení chyb v návrhu a implementaci. Existuje množství technik a přístupů, jakým způsobem software vhodně otestovat. Jedním ze způsobů je manuální a automatické testování.

8.1 Manuální testovaní

Manuální testy jsou prováděny člověkem. Takto byly otestovány části programu, pro které by bylo obtížné vytvořit automatické testy. Jako příklad uveďme otestování editoru elektrických schémat a obecně grafického uživatelského rozhraní (*GUI*). Dále bylo manuální testování použito k finálnímu otestování programu jako celku. Pro otestování dílčích částí programu bylo použito automatické testování.



Obrázek 8.1: Výstup z programu provádějící automatické testy, Zdroj: Vlastní

8.2 Automatické testování

Ukázkový výstup z testovacího programu je na obrázku 8.1. Automatické testování je vhodné pro opakované spouštění velkého množství testů nebo testů s velkým množstvím generovaných dat. Jako software pro automatické testování byl vybrán `QTestLib` framework, což je nástroj pro testovaní aplikací napsaných v Qt. K ověření funkčnosti jednotlivých tříd byla napsána sada desítek testovacích příkladů. Při psaní automatických testů byly uplatněny dva přístupy:

- testování typu černá skříňka (z anglického „black box testing“, známé též jako funkční testování) = testy byly psány bez znalosti implementace. Důraz kladen pouze na vstupy a odpovídající výstupy programu. Odtud vznikl i název pro tuto metodu, kdy nevidíme „dovnitř“ testovaného modulu. Smyslem je otestovat chování programu vzhledem k očekávaným výstupům. Výhodami testování typu černá skříňka jsou rychlost a snadnost, protože test může být spuštěn bez znalosti daného programovacího jazyka. Nevýhodou je riziko, že testovací program nepokryje všechny možné chyby.

- testování typu bílá skříňka (z anglického „white box testing“, známé též jako strukturální testování) = zcela opačný přístup než u testovaní typu černá skříňka. K psaní testů je vyžadována detailní znalost implementace. Cílem je otestovat program tak, aby byly pokryty všechny možné větve programu a hraniční situace. Tento typ testování je obecně mnohem náročnější a nákladnější, protože tester musí kódu porozumět.

Je vhodné oba výše zmíněné přístupy kombinovat. Máme-li k dispozici sadu testovacích příkladů, je snadné po zásahu do programu znovu ověřit jeho funkčnost. Návod na spuštění napsaných testů je součástí přílohy B.4.

Ačkoliv byla aplikace řádně otestována, neznamená to, že je bez chyb. V této souvislosti si dovolím citovat Edsgera W. Dijkstru: *„Testování programu může být velmi efektivním způsobem, jak prokázat přítomnost chyb, ale je naprosto nevhodné k prokázání jejich nepřítomnosti.“*

Kapitola 9

Zhodnocení časové náročnosti

Poslední kapitola poskytuje srovnání časové náročnosti s programem Matlab. Výkonnost programu byla měřena na sadě deseti testovacích obvodů. Testy probíhaly na počítači s procesorem o frekvenci 2,2 GHz a RAM pamětí 2 GB.

9.1 Vyšetřování frekvenčních charakteristik v Matlabu

Způsob vyšetřování charakteristik v Matlabu je zcela odlišný od mého programu. Matlab očekává na svém vstupu tzv. *přenosovou funkci* zkoumaného filtračního článku. Problémem je ovšem její sestavení, protože k tomu Matlab neposkytuje žádné speciální funkce. Proto profesor Erik Cheever z Swarthmore College vytvořil nástroj SCAM[1] (Symbolic Circuit Analysis in MatLab), což je skript pro Matlab sestavující přenosovou funkci k zadanému obvodu. Obvod není bohužel reprezentován graficky, ale textově ve formě tzv. *netlistu*. Netlist je koncept známý ze simulátoru SPICE a definuje vzájemné propojení mezi jednotlivými součástkami.

Uvažujme jednoduchý filtr typu dolní propust s rezistorem a kondenzátorem zapojenými v sérii. Netlist pro tento obvod obsahuje následující údaje:

```
Vin 1 0 1
R1 1 2 1
C1 2 0 1
```

Čísla ve druhém sloupci představují čísla uzlů, ke kterým je součástka připojena zleva. Ve třetím sloupci jsou čísla uzlů, ke kterým je součástka připojena zprava. V posledním sloupci jsou napsány samotné hodnoty součástek.

Z těchto hodnot SCAM sestaví přenosovou funkci. V tomto případě se jedná o funkci:

$$H(s) = \frac{1}{s+1} \tag{9.1}$$

kde s je označení pro úhlovou frekvenci $[rad.s^{-1}]$

Máme-li sestavenou přenosovou funkci, stačí v Matlabu spustit následující posloupnost příkazů. SCAM vrací přenosovou funkci v „textovém“ formátu. Ve skutečnosti až funkce

[1] dostupné z http://www.mathworks.com/matlabcentral/fileexchange/3443-scam-a-tool-for-symbolically-solving-circuit-equations

`tf` vytvoří přenosovou funkci, která je ve vhodném formátu pro funkci `bode`. Napětí `v_2` je mezi uzlem 2 a zemí (uzel 0), odpovídá úbytku napětí na kondenzátoru.

```
>> [n,d] = numden(eval(v_2/Vin));
>> sys = tf(sym2poly(n), sym2poly(d));
>> bode(sys);
```

Pro všechny testovací obvody jsou na přiloženém CD ve složce `matlab` přichystány netlisty. Dále je ve stejné složce pomocný skript, který volá funkce ze SCAM a přímo vykresluje frekvenční charakteristiky v Matlabu.

9.2 Srovnání časové náročnosti s programem Matlab

Na testovací sadě deseti elektrických obvodů z výuky předmětů ITO, IPR a VNV bylo provedeno měření časové náročnosti. Jak již bylo řečeno na začátku této kapitoly, oba programy pracují na zcela odlišných principech. Zatímco navržená aplikace popisuje obvod diferenciálními rovnicemi, Matlab používá již zmíněnou přenosovou funkci.

Naměřené hodnoty jsou uvedeny v tabulce 9.1. Hodnoty všech součástek byly nastaveny na jedna. Rozsah měření nastaven od $0,01\,Hz$ až do $100\,Hz$ (tedy přes 4 dekády). V tabulce jsou pro oba programy uvedeny dva sloupce. V prvním sloupci je celková doba běhu programu, ve sloupci druhém čas spotřebovaný na analýzu obvodu. Tedy čas potřebný k sestavení rovnic nebo přenosové funkce. Výsledky byly zaneseny do grafu, který je na obrázku 9.1.

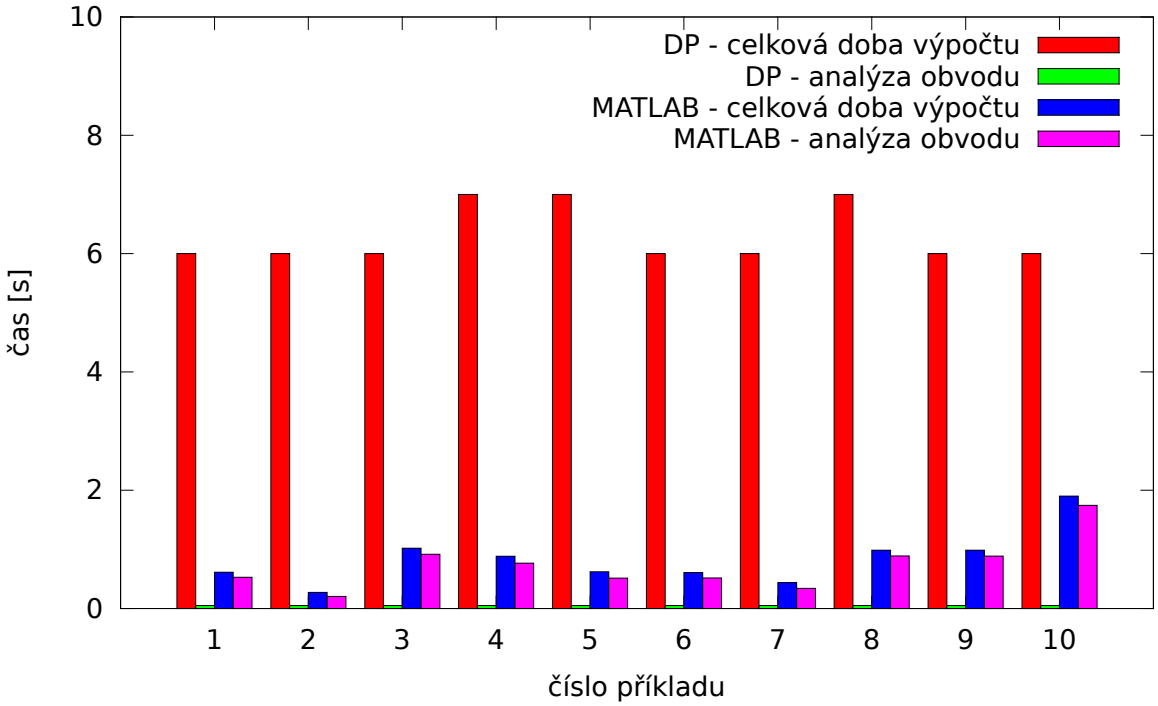
Tabulka 9.1: Srovnání s programem Matlab

Příklad	**DP - doba výpočtu [s]**		**Matlab - doba výpočtu [s]**	
	celkem	**analýza obvodu**	**celkem**	**analýza obvodu**
1	6,284	0,001	0,612	0,528
2	6,346	0,001	0,272	0,204
3	6,494	0,001	1,021	0,916
4	7,322	0,001	0,884	0,767
5	7,289	0,001	0,621	0,513
6	6,184	0,001	0,607	0,516
7	6,420	0,001	0,438	0,340
8	7,588	0,001	0,986	0,889
9	6,657	0,001	0,986	0,886
10	6,560	0,001	1,901	1,745

Zdroj: Vlastní

Protože výpočet v Matlabu je přibližně desetkrát rychlejší, výsledky nejsou pro navržený program zcela příznivé. Je nutné si ale uvědomit důvody. Ačkoliv je řešení diferenciálních rovnic pomocí TKSL/C (simulátor použitý v mém programu) velice rychlé, nemůže konkurovat jednoduchosti výpočtu pomocí přenosové funkce. Výpočet charakteristik z přenosové funkce spočívá pouze v postupném dosazování za frekvenci. Tedy nesrovnatelně jednodušší výpočet. Objektivní srovnání TKSL a Matlabu lze najít v článku [17].

Graf na obrázku 9.1 ukazuje řadu dalších zajímavých výsledků. V případě Matlabu je až 90 % celkové doby výpočtu spotřebováno na sestavení přenosové funkce. Zbývajících 10 % trvá výpočet charakteristiky z přenosové funkce. Oproti tomu sestavení diferenciálních rovnic a obecně analýza obvodu je v navrženém programu cca 500krát rychlejší než pomocí nástroje SCAM v Matlabu.



Obrázek 9.1: Srovnání s programem Matlab, Zdroj: Vlastní

Měření a obecně testování probíhalo samozřejmě na více než deseti obvodech a nebyly zjištěny žádné výrazné časové komplikace. Doba potřebná na proměření jedné dekády se pohybovala přibližně kolem $1,5$ vteřiny (viz tabulka 9.1).

Kapitola 10

Závěr

Simulace na číslicových počítačích je velice rozšířená disciplína. V oblasti elektrotechniky hraje klíčovou roli především v kontrole a ověření správnosti návrhu elektrických a elektronických systémů a to před jejich vlastní výrobou a nasazením. V 50. a 60. letech minulého století byly poprvé představeny techniky pro simulaci obvodů, prvním z rozšířených nástrojů byl program SPICE uvedený roku 1973.

Důležitou vlastností každého obvodu je jeho chování ve frekvenční oblasti. Cílem práce bylo navrhnout a implementovat program, který bude vyšetřovat frekvenční charakteristiky střídavých elektrických obvodů. Výstupem programu jsou frekvenční charakteristiky daného obvodu zachycující závislost přenosu a fáze na frekvenci vstupního signálu. Modelovanými obvody jsou tzv. pasivní kmitočtové filtry. Tedy obvody složeny pouze z pasivních součástek, tj. rezistorů, kondenzátorů a cívek.

Pro popis chování elektrických obvodů jsou použity diferenciální rovnice. Pro jejich řešení byl zvolen systém TKSL založený na modifikované metodě Taylorovy řady a vyznačující se vysokou přesností a rychlostí. V budoucnu bude možné, díky specializovanému paralelnímu systému na čipu FPGA, výpočet v TKSL ještě více urychlit. Nyní bohužel chybí vazba mezi TKSL a paralelním systém. Tato vazba je předmětem aktivního vývoje. Pro navržený program to nebude znamenat žádné dodatečné úpravy. Rozhraní mezi TKSL a výsledným programem zůstane stejné. Akcelerace na FPGA bude řízena v rámci TKSL.

Z měření provedených v kapitole 9 a s následným srovnáním se systémem Matlab vyplynulo, že přístup Matlabu přes tzv. přenosovou funkci je rychlejší než pomocí diferenciálních rovnic. Ačkoliv je Matlab rychlejší, tak i přesto průměrná doba výpočtu $1,5s$ na jednu dekádu není zcela nepříznivá. Překvapivě navržená metoda pro analýzu obvodu a následné sestavení rovnic je výrazně rychlejší než metoda pro sestavení přenosové funkce použita v Matlabu respektive nástavbě SCAM (Symbolic Circuit Analysis in MatLab).

Výhodou řešení pomocí diferenciálních rovnic je možnost vykreslení průběhu napětí na konkrétní součástce, např. kondenzátoru. Dále součástky zapojené v obvodu mohou být i nelineární. Bohužel s tímto přístupem je spojena řada problémů. Největším problémem je tzv. tuhost systému. Což je jev, který vzniká při rozdílných časových konstantách jednotlivých prvků systému. Typicky máme-li děličku napětí s rezistorem a kondenzátorem v sérii a impedance kondenzátoru je mnohem větší než impedance rezistoru. Tento problém je nutné řešit speciálními numerickými metodami. Příznivou skutečností je fakt, že tuhost systému vzniká na pro nás „nezajímavých“ frekvencích. Tedy na frekvencích kdy nedochází ke změně přenosu.

Dalším problémem je ustálení výstupního napětí, kdy kladná i záporná půlvlna v poslední periodě mají stejnou plochu. Budeme-li odečítat amplitudu z neustáleného napětí,

tak se dopustíme chyby. V navrženém programu je nastaven fixní počet deseti period k proměření. Odečítá se amplituda z poslední periody. Experimentálně bylo zjištěno, že tento počet period je dostatečný.

Za hlavní přínos práce bych označil zejména metodu analyzující obvod a automaticky sestavující diferenciální rovnice. Navržená metoda vychází z metod STA a MNA. Nejprve očísluje uzly se stejným potenciálem různými čísly. Pak vytvoří dílčí rovnice pomocí Kirchhoffových zákonů a rovnic součástek. Z těchto rovnic se rekurzivně sestaví výsledná soustava rovnic.

Náplní další práce by mohl být automatický odhad rozsahu frekvencí k proměření a tím by se vyřešil problém s tuhostí systému. Dále by měla být lepším způsobem prováděna kontrola ustálení výstupního napětí. Posledním možným vylepšením by byl přesnější odečet amplitudy. Amplituda by se odečítala normálním způsobem jako doposud, kolem této amplitudy by se spustila simulace znovu s menším krokem.

Program byl důsledně otestován a je plně funkční na operačních systémech Windows a Linux (distribuce CentOS). Výsledky práce mohou být využity při výuce v předmětech ITO, IPR a VNV.

Literatura

[1] BLAHOVEC, A.: *Elektrotechnika II.* Praha: Informatorium, Čtvrté vydání, 2003, ISBN 80-7333-013-X.

[2] BLANCHETTE, J.; SUMMERFIELD, M.: *C++ GUI programming with Qt 4.* Courier in Stoughton, Massachusetts, první vydání, 2006, ISBN 0-13-187249-4.

[3] DOSTÁL, T.; AXMAN, V.: Elektrické filtry. *Vysoké učení technické VUT*, 2004.

[4] FAJMON, B.; RŮŽIČKOVÁ, I.: Matematika 3. *Vysoké učení technické VUT*, 2005.

[5] HACHTEL, G. D.; BRAYTON, R. K.; GUSTAVSON, F. G.: The Sparse Tableau Approach to Network Analysis and Design. 1971.

[6] HANKE, M.: An Introduction to the Modified Nodal Analysis. 2006.

[7] HO, C.-W.; RUEHLI, A. E.; BRENNAN, P. A.: The Modified Nodal Approach to Network Analysis. 1975.

[8] HÁJEK, K.; SEDLÁČEK, J.: *Kmitočtové filtry.* Praha: Nakladatelství BEN - technická literatura, první vydání, 2002, ISBN 80-7300-023-7.

[9] JANKO, R.: *Simulátor střídavých elektronických obvodů.* Bakalářská práce, FIT VUT v Brně, Brno, 2011.

[10] JANKO, R.: *Modelování elektrických obvodů ve specializovaném paralelním systému.* Semestrální projekt, FIT VUT v Brně, Brno, 2013.

[11] KRAUS, M.: *Paralelní výpočetní architektury založené na numerické integraci.* Disertační práce, FIT VUT v Brně, Brno, 2013.

[12] KUNOVSKÝ, J.: *Modern Taylor Series Method.* Habilitační práce, 1995.

[13] KUNOVSKÝ, J.: *Prvky počítačů.* Vysoké učení technické VUT, 2006.

[14] KUNOVSKÝ, J.; KRAUS, M.; VOPĚNKA, V.: Runge-Kutta Based Parallel Computations. *7th Vienna Conference on Mathematical Modelling*, 2012.

[15] KUNOVSKÝ, J.; KRAUS, M.; ŠÁTEK, V.; aj.: Parallel Computations Based on Analogue Principles. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation*, IEEE Computer Society, 2009, ISBN 978-0-7695-3593-7, s. 111–116.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=8889

[16] KUNOVSKÝ, J.; KRAUS, M.; ŠÁTEK, V.; aj.: Parallel Computations Based on Numerical Integration Methods. In *Proceedings of the 10th International Conference of Numerical Analysis and Applied Mathematics*, 1472, American Institute of Physics, 2012, ISBN 978-0-7354-1091-6, ISSN 1551-7616, str. 4.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10089

[17] KUNOVSKÝ, J.; ŠÁTEK, V.; KRAUS, M.; aj.: Comparison of TKSL to world standards. *International Journal of Autonomic Computing*, ročník 1, č. 2, 2009: s. 182–191, ISSN 1741-8569.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=8937

[18] LÁNÍČEK, R.: *Elektronika: Obvody, součástky, děje*. Praha: Nakladatelství BEN - technická literatura, první vydání, 1998, ISBN 80-86056-25-2.

[19] MURINA, M.: *Teorie obvodů*. VUTIUM Brno, 2000.

[20] NAJM, F. N.: *Circuit simulation*. Hoboken, New Jersey: John Wiley & Sons, 2010, ISBN 978-0-470-53871-5.

[21] PERINGER, P.: Modelování a simulace. *Vysoké učení technické VUT*, 2008.

[22] PIETRIK, M.: *Automatizované testování webových aplikací*. Diplomová práce, Fakulta informatiky Masarykovy univerzity, Brno, 2012.

[23] Qt: Webové stránky frameworku Qt [online]. 2013.
URL http://http://qt.digia.com/

[24] RAGHURAM, R.: *Computer Simulation Of Electronic Circuits*. New Delhi: Wiley Eastern Limited, 1991, ISBN 81-224-0111-2.

[25] RÁBOVÁ, Z.: *Modelování a simulace*. Vysoké učení technické VUT, 1992, ISBN 80-214-0480-9.

[26] TKSL/C: Webové stránky o TKSL/C [online]. 2013.
URL http://www.fit.vutbr.cz/~kunovsky/TKSL/tkslc.html.cs

[27] URBÁNEK, R.: *Frekvenční charakteristiky*. Diplomová práce, FIT VUT v Brně, Brno, 2006.

Dodatek A

Obsah CD

Přiložené CD obsahuje následující adresáře:

- **bin** - adresář se spustitelným souborem pro Windows.
- **matlab** - adresář se soubory pro vyšetření frekvenčních charakteristik v Matlabu.
- **src** - adresář se zdrojovými kódy a testovacími soubory.
- **zprava** - adresář s textem diplomové práce.

Dodatek B

Manuál k programu

Program vykresluje k zapojenému obvodu frekvenční charakteristiky. Kapitola popisuje, jak vytvořený program přeložit a ovládat. A dále jakým způsobem spustit sadu testů, která testuje funkčnost aplikace.

B.1 Požadavky

Pro Linux

- Qt framework (testováno pro verzi 4.8.0)
- překladač g++ (testováno pro verzi 4.7.2)
- TKSL/C v adresáři s přeloženým projektem

Pro Windows (při překladu)

- Qt framework (testováno pro verzi 4.7.3-2)
- překladač MinGW (testováno pro verzi 5.1.6)
- TKSL/C v adresáři s přeloženým projektem

V případě Windows není nutné překládat projekt ani mít nainstalovaný Qt framework. Na přiloženém CD ve složce `bin` je přichystán přeložený program i s potřebnými knihovnami z Qt.

B.2 Překlad

Je možné otevřít přiložený `.pro` v Qt Creatoru a tam projekt spustit. Nebo program přeložit v příkazovém řádku:

1. V souboru `DIP.pro` zakomentovat `test_conf` v `CONFIG`:

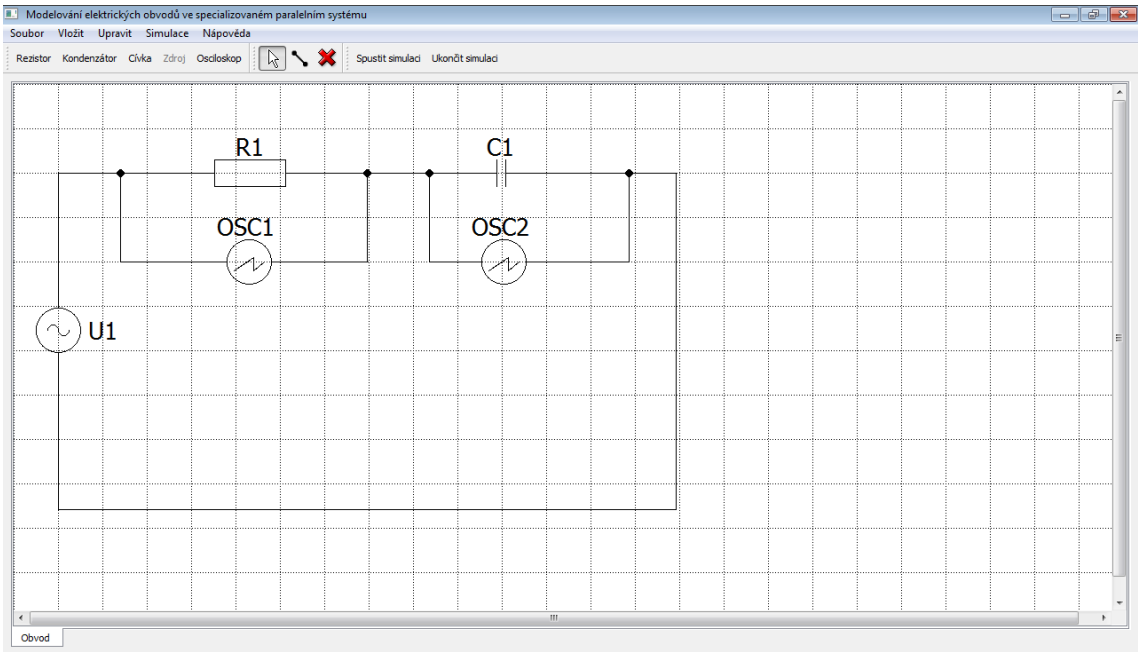
   ```
   CONFIG += static #test_conf
   ```

2. Vygenerovat Makefile (příkazem qmake-qt4 nebo qmake):

   ```
   $ qmake-qt4 -makefile
   ```

3. Přeložit a spustit:

```
$ make
$ ./DIP
```



Obrázek B.1: Editor elektrických schémat, Zdroj: Vlastní

B.3 Ovládání a popis programu

Vložení součástek do schématu - do schématu (viz obrázek B.1) lze vložit rezistory, kondenzátory, cívky, zdroje a osciloskopy. Součástky můžu vložit z toolbaru nebo přes menu, záložka „Vložit“.

Propojení součástek - součástky se propojují vodičem. Vodič opět můžeme vložit přes menu nebo toolbar. Další možností je klávesová zkratka „Ctrl-V“. Vodič se začne přidávat až po kliknutí na součástku nebo jiný vodič, který chci propojit. Přidávání vodiče lze ukončit ukončit stiskem „Esc“ nebo po kliknutí na součástku nebo vodič.

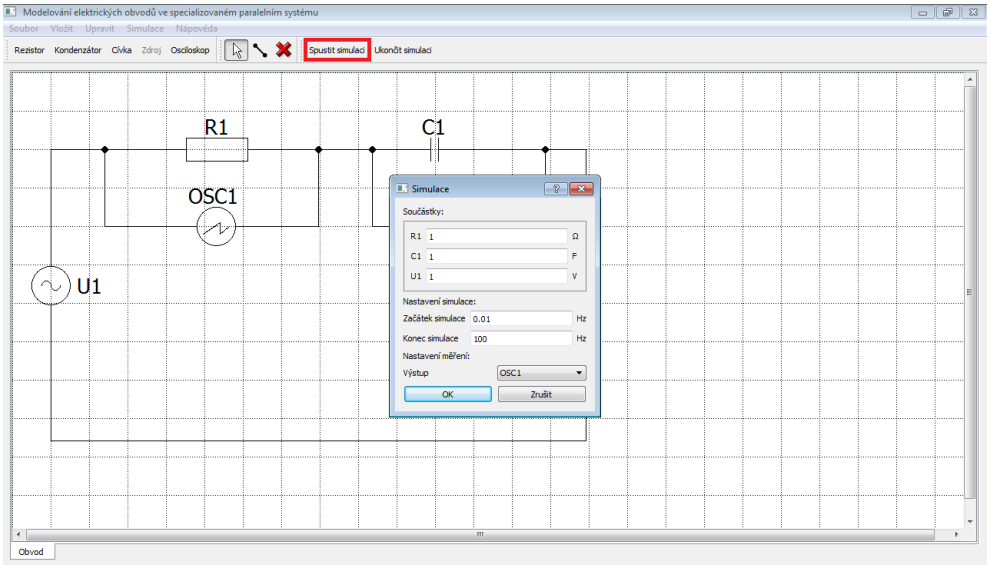
Odstranění součástky nebo vodiče - kliknutím na součástku ji vyberu a klávesou „Delete“ odstraním. Akce pro odstranění je opět přístupná přes menu nebo toolbar.

Nastavení hodnot součástek - dvojklikem myší na součástku se otevře dialogové okno. Okno můžeme také otevřít přes menu „Upravit“ → „Vlastnosti součástky“, ale musí být vybrána nějaká součástka.

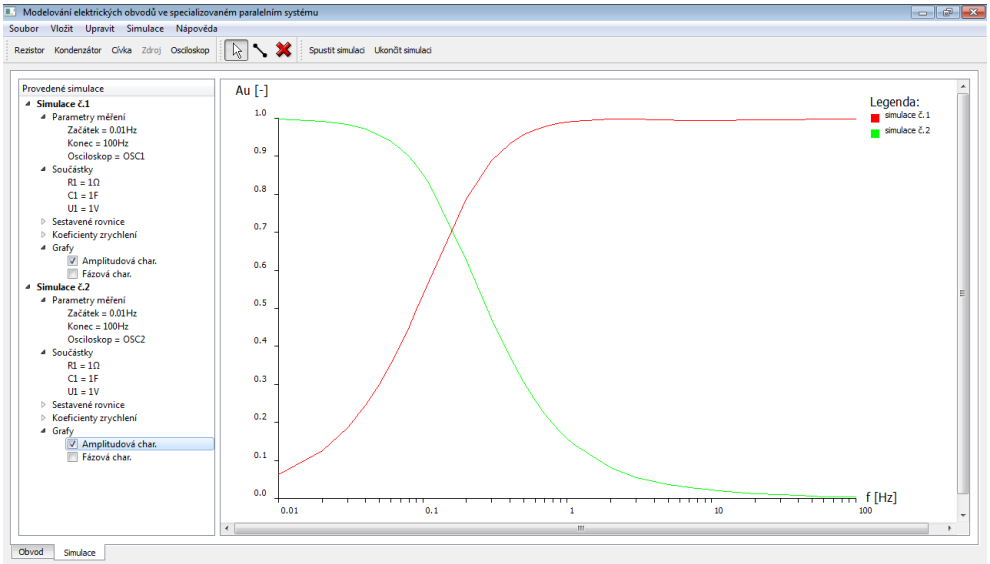
Spuštění simulace - kliknutím na „Spustit simulaci“ v toolbaru (viz obrázek B.2) se uživateli zobrazí okno, kde může změnit parametry součástek, musí nastavit rozsah měření a vybrat osciloskop, na kterém se bude měřit výstupní napětí. Stiskem tlačítka „OK“ se simulace spustí. Během jednotlivých simulací lze měnit zapojení obvodu.

Zobrazení výsledků - v průběhu simulace se program přepne na kartu „Simulace“. Po dokončení simulace se do výpisu vlevo přidá položka (viz obrázek B.3). Ve výpisu ke každé simulaci jsou uloženy všechny potřebné informace jako parametry měření, hodnoty součástek, použité rovnice, koeficienty zrychlení. Ve výpisu je dále možné zaškrtnutím položky „Grafy“ si zobrazit výsledky.

Ukončení simulace - kliknutím na „Ukončit simulaci“ v toolbaru. Výsledky simulací budou vymazány.



Obrázek B.2: Nastavení parametrů simulace a její spuštění, Zdroj: Vlastní



Obrázek B.3: Výsledky simulace, Zdroj: Vlastní

B.4 Použití testovacího programu

K aplikaci je vytvořena sada testů. Jedná se o konzolovou aplikaci, pro spuštění je potřeba postupovat následovně.

1. V souboru `DIP.pro` odkomentovat `test_conf` v `CONFIG`:

   ```
   CONFIG += static test_conf
   ```

2. Vygenerovat nový Makefile (příkazem `qmake-qt4` nebo `qmake`):

   ```
   $ qmake-qt4 -makefile
   ```

3. Přeložit a spustit:

   ```
   $ make
   $ ./DIP_testy
   ```